TUGAS AKHIR – KI141502

PATHFINDING MENGGUNAKAN ALGORITMA DIJKSTRA PADA GAME 'AVIAR' MULTIPLAYER BERBASIS VIRTUAL REALITY

VINSENSIA SIPRIANA ZEGA
NRP 05111440000066

Dosen Pembimbing I
Dr. Eng. Darlis Herumurti, S.Kom., M.Kom.

Dosen Pembimbing II
Dini Adni Navastara, S.Kom., M.Sc.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

TUGAS AKHIR – KI141502

PATHFINDING MENGGUNAKAN ALGORITMA DIJKSTRA PADA GAME 'AVIAR' MULTIPLAYER BERBASIS VIRTUAL REALITY

VINSENSIA SIPRIANA ZEGA
NRP 05111440000066

Dosen Pembimbing I
Dr. Eng. Darlis Herumurti, S.Kom., M.Kom.

Dosen Pembimbing II
Dini Adni Navastara, S.Kom., M.Sc.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2018

[*Halaman ini sengaja dikosongkan*]

FINAL PROJECT – KI141502

PATHFINDING USING DIJKSTRA ALGORITHM ON 'AVIAR', MULTIPLAYER GAME BASED ON VIRTUAL REALITY

VINSENSIA SIPRIANA ZEGA
NRP 05111440000066

Supervisor I
Dr. Eng. Darlis Herumurti, S.Kom., M.Kom.

Supervisor II
Dini Adni Navastara, S.Kom., M.Sc.

DEPARTMENT OF INFORMATICS
Faculy of Information and Communication Technology
Institut Teknologi Sepuluh Nopember
Surabaya, 2018

[*Halaman ini sengaja dikosongkan*]

LEMBAR PENGESAHAN

PATHFINDING MENGGUNAKAN ALGORITMA DIJKSTRA PADA GAME 'AVIAR' MULTIPLAYER BERBASIS VIRTUAL REALITY

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Rumpun Mata Kuliah Interaksi Grafika dan Seni
Program Studi S-1 Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:
VINSENSIA SIPRIANA ZEGA
NRP: 05111440000066

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Dr. Eng. Darlis Herumurti, S.Kom., M.Kom.
NIP. 197712172003121001 ..... (Pembimbing 1)

Dini Adni Navastara, S.Kom., M.Sc.
NIP. 198510172015042001 ..... (Pembimbing 2)

SURABAYA
JULI, 2018

[Halaman ini sengaja dikosongkan]

PATHFINDING MENGGUNAKAN ALGORITMA DIJKSTRA PADA GAME 'AVIAR' MULTIPLAYER BERBASIS VIRTUAL REALITY

Nama Mahasiswa : Vinsensia Sipriana Zega
NRP : 05111440000066
Jurusan : Informatika, FTIK ITS
Dosen Pembimbing 1 : Dr. Eng. Darlis Herumurti, S.Kom., M.Kom.
Dosen Pembimbing 2 : Dini Adni Navastara, S.Kom., M.Sc.

ABSTRAK

Seiringnya perkembangan zaman, salah satu hiburan yang bisa didapat berhubungan erat dengan perkembangan teknologi komputer. Salah satunya ialah aplikasi permainan dengan menggunakan sistem Virtual Reality (VR) dan penerapan Artificial Intelligence (AI), sebagai contohnya yaitu permainan labirin. Dengan menggunakan sistem VR, permainan labirin akan terlihat lebih nyata. Lalu AI pathfinding akan diimplementasikan pada musuh untuk meningkatkan kesulitan pada permainan.Tugas akhir ini mengusulkan penerapan AI pathfinding menggunakan algoritma Dijkstra dalam permainan labirin berbasis virtual reality.

Algoritma dijksra akan mencari jalan dengan cost terkecil antara musuh terhadap pemain. Pada tugas akhir ini Artifical Intelligence akan dibagi menjadi 4 tingkat kesulitan yaitu easy, medium, hard, dan insane. Dimana musuh akan memiliki kesempatan untuk bergerak secara acak yang berbeda-beda yaitu easy 50%, medium 30%, hard 15%, dan insane 0%.

Uji coba dilakukan terhadap masing-masing tingkat kesulitan pada labirin berukuran 8x8 dengan kondisi awal yang sama. Berdasarkan hasil uji coba, metode Dijkstra Pathfinding dapat menemukan jalan tercepat dengan waktu pada setiap tingkatannya yaitu easy 230,26 detik, medium 164,53 detik, hard 135,53 detik dan insane 121,14 detik.

Kata kunci: Dijkstra Pathfinding, Artificial Intelligence, Permainan Labirin, Virtual Reality, Oculus Rift.

PATHFINDING USING DIJKSTRA ALGORITHM ON 'AVIAR', MULTIPLAYER GAME BASED ON VIRTUAL REALITY

Student Name	: Vinsensia Sipriana Zega
Registration Number	: 05111440000066
Department	: Informatics Engineering, FTIK ITS
First Supervisor	: Dr. Eng. Darlis Herumurti, S.Kom., M.Kom.
Second Supervisor	: Dini Adni Navastara, S.Kom., M.Sc.

ABSTRACT

As the times progressed, one of the entertainment that can be obtained is closely related to computer technology development. One of them is a game application using Virtual Reality (VR) system and the implementation of Artificial Intelligence (AI), a maze game for example. By using the VR system, the maze will looks and feels more real. And then AI Pathfinding will be implemented to the enemy to increase the difficulty of the game. This research proposes AI pathfinding implementation using Dijkstra Algorithm in the virtual reality based maze game.

Dijkstra algorithm will find the lowest cost path between enemy against player. In this research, the artificial intelligence will be divided into 4 different level of difficulties namely easy, medium, hard, and insane. Where the enemy will have different chances to move randomly that is to say easy 50%, medium 30%, hard 15%, and insane 0%.

The experimental is done to each difficulties in an 8x8 maze with similar starting condition. Based on the experiment, Dijkstra pathfinding method is able to find the fastest path with average time for each difficulties are easy 230,26 seconds, medium 164,53 seconds, hard 135,53 seconds and insane 121,14 seconds.

Keywords: Dijkstra Pathfinding, Artificial Intelligence, Maze Game, Virtual Reality, Oculust Rift.

KATA PENGANTAR

Puji Syukur kepada Tuhan Yang Maha Esa, atas berkatNya penulis dapat menyelesaikan Tugas Akhir yang berjudul:

"**PATHFINDING MENGGUNAKAN ALGORITMA DIJKSTRA PADA GAME 'AVIAR' MULTIPLAYER BERBASIS VIRTUAL REALITY"**

Selain ini, pada kesempatan ini penulis menghanturkan terima kasih sebesar-besarnya kepada pihak-pihak yang tanpa mereka, penulis tidak dapat menyelesaikan buku ini:

1. Yesus Kristus - atas segala limpahan rahmat dan karunia-Nya penulis dapat menyelesaikan Tugas Akhir dan juga perkuliahan di Departemen Informatika ITS .
2. Keluarga penulis, yang tiada hentinya memberikan dukungan doa, nasihat, dan selalu mengingatkan penulis agar tidak lupa menjaga kesehatan saat menyelesaikan Tugas Akhir ini.
3. Bapak Dr. Eng. Darlis Herumurti, S.Kom., M.Kom. dan Ibu Dini Adni Navastara, S.Kom., M.Sc. selaku pembimbing I dan II yang telah membimbing dan memberikan motivasi, nasihat dan bimbingan dalam menyelesaikan Tugas Akhir ini.
4. Bapak dan Ibu dosen Departemen Informatika ITS yang telah banyak memberikan ilmu dan bimbingan yang tak ternilai harganya bagi penulis.
5. Kedua teman kelompok tugas akhir penulis, Aufar Rizqi dan Anggit Yudhistira yang telah membantu dan memberikan semangat kepada penulis.
6. Ghaly Aditya, Muh. Ghazian dan Firman Maulana yang sudah membantu dan mau direpotin penulis untuk menjadi pengajar selama pengerjaan tugas akhir.

7. Rekan-rekan di Laboratorium IGS dan “Lodgiwetan”, yang selalu memberikan semangat dalam hal studi maupun masalah lainnya
8. Sahabat penulis : Anindita, Pertiwi, Uly, Rara, Elva, Nafia, Bimo, Shafly (Om), Glleen, Afif, Rage, Aldo, Rifat, Faishal, Dwika, Ivaldy, Aditya Gunawan, Riansya, Deni, Farhan, Aditya Ikhsan, Tosca, Resha yang sudah memberikan banyak pengalaman dan canda tawa selama masa perkuliahan
9. Teman-teman TC 2014, kakak-kakak TC 2012 & 2013 dan adik-adik TC 2015 & 2016 yang tidak dapat disebutkan satu per satu yang selalu membantu, menghibur, menjadi tempat bertukar ilmu serta pembelajaran baru dan berjuang bersama-sama penulis.
10. Serta semua pihak yang yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari masih ada kekurangan dalam penyusunan tugas akhir ini. Penulis mohon maaf atas kesalahan, kelalaian maupun kekurangan dalam penyusunan tugas akhir ini. Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan ke depan.

Surabaya, Juli 2018

Vinsensia Sipriana Zega

DAFTAR ISI

ABSTRAK ..... **vii**
ABSTRACT ..... **ix**
KATA PENGANTAR ..... **xi**
DAFTAR ISI ..... **xiii**
DAFTAR GAMBAR ..... **xvii**
DAFTAR TABEL ..... **xix**
DAFTAR KODE SUMBER ..... **xxi**
BAB I PENDAHULUAN **1**
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Batasan Masalah 2
1.4 Tujuan Tugas Akhir 3
1.5 Manfaat Tugas Akhir 3
1.6 Metodologi 3
1.7 Sistematika Laporan 4
BAB II DASAR TEORI **7**
2.1 *Virtual Reality* 7
2.2 Oculus Rift 7
2.3 *Artificial Intelligence* 8
2.4 *Pathfinding* 8
2.5 Algoritma Dijkstra 9
2.6 Unity (Game Engine) 10
2.7 Bahasa Pemrograman C# 10
2.8 Mixamo.com 11
BAB III ANALISIS DAN PERANCANGAN SISTEM **13**
3.1 Analisis Sistem 13
3.1.1 Spesifikasi Kebutuhan Sistem 13
3.2 Perancangan Sistem 14
3.2.1 Deskripsi Umum Sistem 14
3.2.2 Arsitektur Sistem 14
3.3 Penerapan *Artificial Intelligence* 15

3.3.1 Penerapan *Node* sebagai *Path* menggunakan Model *Maze*15
3.3.2 Penerapan *Artificial Intelligence* menggunakan Algoritma Dijkstra 16
3.3.3 Penerapan *Artificial Intelligence* pada tingkat kesulitan *game* 19
BAB IV IMPLEMENTASI.....23
4.1 Lingkungan Implementasi.....23
4.2 Implementasi *Import Asset* dan *Package* 23
4.3 Implementasi Algoritma Djikstra *Pathfinding* 25
4.4 Implementasi pada Musuh 27
4.4.1 Implementasi *health* pada Musuh.....27
4.4.2 Implementasi *movement* pada musuh 28
BAB V UJI COBA DAN EVALUASI.....37
5.1 Lingkungan Uji Coba.....37
5.2 Pengujian Fungsionalitas.....37
5.2.1 Uji Coba Musuh Muncul Dalam *Maze*.....38
5.2.2 Uji Coba Musuh Mengetahui Kondisi *Node* pada *Maze* ...39
5.2.3 Uji Coba Musuh Mengetahui Posisi Pemain 42
5.2.4 Uji Coba Musuh Mencari Jalan Tercepat Menuju Pemain 44
5.2.5 Uji Coba Musuh Menyerang Pemain.....47
5.2.6 Uji Coba Efektivitas Algoritma Dijkstra 48
5.2.7 Hasil Uji Coba 49
5.3 Pengujian Pengguna 49
5.3.1 Skenario Pengujian Pengguna 50
5.3.2 Daftar Penguji Permainan.....54
5.3.3 Hasil Pengujian Pengguna 54
5.3.4 Kritik dan Saran Pengguna 55
5.4 Evaluasi Pengujian.....56
BAB VI KESIMPULAN DAN SARAN 59
6.1 Kesimpulan 59
6.2 Saran.....59
DAFTAR PUSTAKA 61
LAMPIRAN.....63

BIODATA PENULIS.....77

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

Gambar 2.1 Beberapa komponen Oculus Rift.....8
Gambar 3.1 Arsitektur Sistem.....15
Gambar 3.2 Diagram Alir Algoritma Dijkstra *Pathfinding*.....16
Gambar 3.3 Kondisi awal pada *maze*.....17
Gambar 3.4 Langkah awal.....17
Gambar 3.5 *Trace Initial State* ke *Goal State*.....18
Gambar 3.6 *Solution Path*.....18
Gambar 3.7 Musuh pada tingkat kesulitan *easy*.....19
Gambar 3.8 Musuh pada tingkat kesulitan *medium*.....20
Gambar 3.9 Musuh pada tingkat kesulitan *hard*.....21
Gambar 3.10 Musuh pada tingkat kesulitan *insane*.....21
Gambar 4.1 Implementasi *Import Asset*.....24
Gambar 5.1 Kondisi awal *maze*.....38
Gambar 5.2 Hasil Uji Coba Musuh Mengetahui Kondisi *Maze*..41
Gambar 5.3 Ilustrasi *neighbour* dari *node*.....41
Gambar 5.4 Matriks *Adjacency Node*.....42
Gambar 5.5 Graf *Adjacency Node*.....42
Gambar 5.6 Hasil Uji Coba Musuh Mengetahui Posisi Pemain..44
Gambar 5.7 Hasil Uji Coba Musuh Mencari Jalan Menuju Pemain
.....46

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

Tabel 3.1 Kebutuhan Fungsional Sistem.....13
Tabel 3.2 Kebutuhan Non-Fungsional Sistem.....13
Tabel 3.3 Pembagian tingkat kesulitan *game*.....19
Tabel 4.1 Lingkungan Implementasi.....23
Tabel 5.1 Lingkungan Pengujian Sistem.....37
Tabel 5.2 Hasil Uji Coba Musuh Muncul Dalam Maze.....38
Tabel 5.3 Hasil Uji Coba Musuh Mengetahui Kondisi *Node* pada *Maze*.....39
Tabel 5.4 Hasil Uji Coba Musuh Mengetahui Posisi Pemain.....42
Tabel 5.5 Hasil Uji Coba Musuh Mencari Jalan Tercepat Menuju Pemain.....45
Tabel 5.6 Hasil Uji Coba Musuh Menyerang Pemain.....47
Tabel 5.7 Hasil Uji Coba Efektivitas Algoritma Dijkstra Tingkat Kesulitan Berdasarkan Waktu.....48
Tabel 5.8 Hasil Uji Coba.....49
Tabel 5.9 Rentang Nilai.....50
Tabel 5.10 Kuesioner Pengguna.....50
Tabel 5.11 Daftar Penguji Permainan.....54
Tabel 5.12 Hasil Pengujian Pengguna.....55
Tabel 5.13 Hasil Akhir Pengujian Pengguna.....55
Tabel 5.14 Kritik dan Saran Pengguna.....56

[*Halaman ini sengaja dikosongkan*]

DAFTAR KODE SUMBER

Kode Sumber 4.1 Implementasi Algoritma Dijkstra *Pathfinding*27
Kode Sumber 4.2 Implementasi *health* pada musuh 28
Kode Sumber 4.3 Implementasi *movement* pada Musuh 30
Kode Sumber 4.4 Implementasi musuh pada tingkat kesulitan *easy*.....31
Kode Sumber 4.5 Implementasi musuh pada tingkat kesulitan *medium* 33
Kode Sumber 4.6 Implementasi musuh pada tingkat kesulitan *hard* 35
Kode Sumber 4.7 Implementasi musuh pada tingkat kesulitan *insane* 36

[*Halaman ini sengaja dikosongkan*]

BAB I
PENDAHULUAN

Pada bab ini dibahas hal-hal yang mendasari tugas akhir. Bahasan meliputi latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika laporan tugas akhir.

1.1 Latar Belakang

Saat ini kata teknologi komputer sangat erat dengan kegiatan manusia. Salah satu bentuk teknologi komputer yang sedang *booming* ialah *virtual reality*. *Virtual reality* merupakan teknologi yang dapat membangkitkan suasana tiga dimensi (*3D*) atau nyata yang membuat penggunanya seolah-olah berada di dunia nyata meskipun sebenarnya bukan berada di dunia nyata [1]. *Virtual reality* sudah banyak diterapkan di dunia permainan (game).

Permainan (*game*) merupakan suatu kegiatan menyenangkan yang dapat dilakukan oleh siapapun dan dimanapun dengan tujuan untuk menghibur dan mengisi waktu luang. *Game* sangat bermanfaat dalam perkembangan otak dan membantu melatih kemampuan pemain untuk dapat menyelesaikan berbagai permasalahan yang muncul dalam *game*. Salah satu tipe permainan yang sering dimainkan banyak kalangan ialah permainan labirin (*game maze*). Permainan labirin (*game maze*) adalah tipe permainan yang bertujuan untuk menemukan jalan keluar diantara jalan yang bercabang dan jalan buntu. Permainan ini sangat membutuhkan konsentrasi dalam menyelesaikan suatu masalah yang ada. Pemain mencari jalan keluar sambil menghindari serangan dari musuh yang berada dalam game. Serangan dari musuh merupakan salah satu contoh dari *artificial intelligence* [3].

Pengaruh sebuah *artificial intelligence* dapat membuat sebuah permainan menjadi lebih menarik atau tidak menarik untuk dimainkan [4]. Karena selain dapat membuat musuh lebih pintar, *artificial intelligence* dapat membuat pemain menjadi lebih

cerdas dalam berfikir karena jalan dan taktik yang digunakan musuh selalu berubah-ubah. *Artificial intelligence* yang sering ditemui dalam permainan maze ialah *pathfinding* yang digunakan sebagai pencarian jalan pada *maze*.

Penelitian ini membahas tentang *pathfinding* dalam permainan 'AVIAR' yang merupakan sebuah permainan labirin (*maze*) berbasis *virtual reality*, dimana pemain diminta untuk mencari jalan keluar dari labirin dengan waktu yang ditentukan. Apabila pemain tidak bisa melewati labirin dengan waktu yang telah ditentukan, maka bentuk labirin akan berubah dan akan ada serangan dari musuh yang muncul dalam labirin. Untuk mempermudah musuh mencari jalan terpendek menyerang pemain, maka akan digunakan *pathfinding*.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana cara musuh mengetahi posisi dari pemain?
2. Bagaimana cara musuh mendapatkan jalan tercepat ke pemain?
3. Bagaimana cara mengetahui algoritma Dijkstra efektif dalam permainan?

1.3 Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan antara lain:

1. Aplikasi yang dibuat merupakan aplikasi dekstop.
2. Lingkungan pengembangan yang digunakan menggunakan aplikasi Unity 2017.3.0f3 Free License dan Bahasa Pemrograman C#.
3. Menggunakan Oculus Rift SDK.
4. Menggunakan algoritma Dijkstra sebagai *pathfinding*.
5. Musuh hanya muncul pada malam hari.
6. Terdapat 4 musuh yang dibedakan sesuai tingkat kesulitan *game*.

7. Permainan akan berakhir apabila *health bar* pemain habis.

1.4 Tujuan Tugas Akhir

Tujuan tugas akhir ini adalah mengimplementasikan algoritma Dijkstra *Pathfinding* dalam *game* ‘AVIAR’.

1.5 Manfaat Tugas Akhir

Manfaat dari tugas akhir ini adalah untuk membantu melatih kemampuan pemain untuk dapat menyelesaikan berbagai permasalahan yang muncul pada *game* ‘AVIAR’ *multiplayer* berbasis *virtual reality*.

1.6 Metodologi

Tahapan-tahapan yang dilakukan dalam pengerjaan tugas akhir ini adalah sebagai berikut:

1. Studi Literatur
   Pada tahap studi literatur ini, penulis akan mempelajari beberapa referensi terkait topik tugas akhir. Studi literatur ini juga memiliki beberapa sumber referensi seperti buku, internet, ataupun materi dalam suatu mata kuliah yang berhubungan dengan topik tugas akhir ini. Beberapa referensi yang dipelajari ialah mengenai Unity, Oculus Rift Controller. Oculus Rift SDK, dan algoritma Dijkstra.
2. Analisis dan Desain Perangkat Lunak
   Pada tahap ini akan dilakukan analisa perencanaan dan pendefinisian kebutuhan sistem untuk mengetahui permasalahan yang akan dihadapi pada tahap implementasi. Selanjutnadijabarkan kebutuhan-kebutuhan tersebut kedalam perancangan fitur pada sistem.

3. Implementasi Perangkat Lunak
   Pembangunan aplikasi ini akan dibangun menggunakan Game Engine Unity 3D Free, dengan Bahasa Pemrograman C#.

Aplikasi ini juga menggunakan Oculus Rift SDK yang akan dihubungkan dengan Oculus Rift Controller.

4. Pengujian dan Evaluasi

   Tahap Pengujian dan Evaluasi berisi pengujian aplikasi dan evaluasi berdasarkan hasil pengujian. Pengujian akan dilakukan oleh sampel pengguna, yaitu mahasiswa Teknik Informatika ITS sebagai Sampel dari masyarakat indonesia pada umumnya.

1.7 Sistematika Laporan

Buku tugas akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan tugas akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku tugas akhir terdiri atas beberapa bagian seperti berikut:

Bab I Pendahuluan

Bab yang berisi mengenai latar belakang, tujuan, dan manfaat dari pembuatan tugas akhir. Selain itu permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penulisan juga merupakan bagian dari bab ini.

Bab II Dasar Teori

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan tugas akhir ini.

Bab III Analisis dan Perancangan Sistem

Bab ini berisi tentang analisis dan perancangan desain sistem tugas akhir ini.

Bab IV Implementasi

Bab ini membahas implementasi dari desain yang telah dibuat pada bab sebelumnya. Penjelasan berupa kode yang digunakan untuk proses implementasi.

Bab V Pengujian dan Evaluasi

Bab ini membahas tahap-tahap uji coba. Kemudian hasil uji coba dievaluasi untuk kinerja dari aplikasi yang dibangun.

Bab VI Kesimpulan dan Saran

Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan aplikasi ke depannya.

Daftar Pustaka

Merupakan daftar referensi yang digunakan untuk mengembangkan Tugas Akhir.

Lampiran

Merupakan bab tambahan yang berisi daftar istilah yang penting pada aplikasi ini.

[Halaman ini sengaja dikosongkan]

BAB II
DASAR TEORI

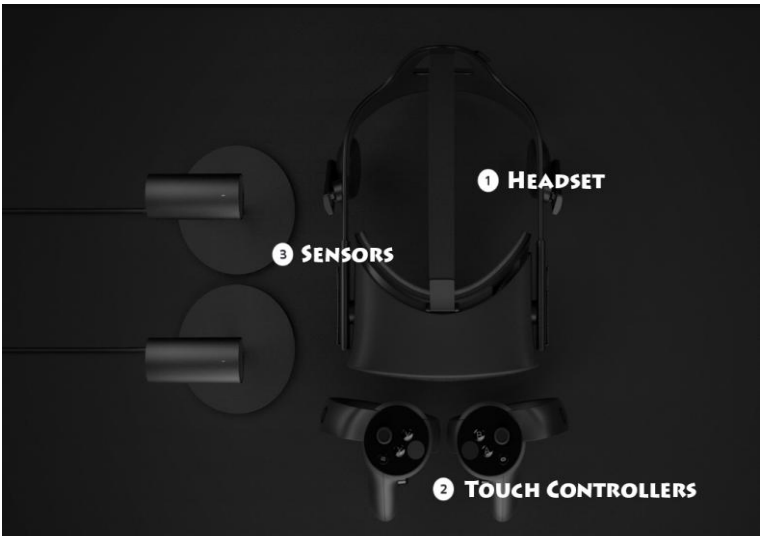
Pada bab ini diuraikan mengenai dasar-dasar teori yang digunakan dalam pengerjaan tugas akhir dengan tujuan untuk memberikan gambaran secara umum terhadap penelitian yang dikerjakan.

2.1 *Virtual Reality*

Virtual reality merupakan teknologi yang menampilkan dunia *virtual* tiga dimensi yang disimulasikan oleh komputer. *Virtual reality* sangat merekat pada dunia grafika komputer. Pengalaman pengguna dengan menggunakan *virtual reality* dapat direndam secara efektif di dunia *virtual* yang responsif. Untuk menciptakan sensasi tersebut diperlukan beberapa perangkat pendukung berupa *headmounth*, *headset*, *walker*, *suit* dan *glove*. Para ahli menciptakan teknologi ini bertujuan untuk membantu pekerjaan manusia menjadi lebih mudah di berbagai bidang seperti kedokteran, arsitek, pendidikan dan lain sebagainya [1].

2.2 Oculus Rift

Oculus Rift merupakan *headset virtual reality* baru yang dirancang khusus untuk *video game* yang akan mengubah cara anda berpikir tentang game selamanya. Dengan bidang pandang yang sangat luas, layar resolusi tinggi, dan pelacakan kepala latensi ultra-rendah, Rift memberikan pengalaman yang benar-benar imersif yang memungkinkan anda untuk masuk ke dalam permainan favorit anda dan menjelajahi dunia baru yang belum pernah ada sebelumnya [2]. Beberapa komponen dari Oculus Rift dapat dilihat pada **Gambar 2.1**.



Gambar 2.1 Beberapa komponen Oculus Rift

2.3 *Artificial Intelligence*

Artificial Intelligence merupakan *intelligence* yang ditambahkan kepada suatu sistem yang bisa diatur dalam konteks ilmiah atau *Artificial Intelligence* dapat didefinisikan sebagai kecerdasan entitas ilmiah. Sistem seperti ini umumnya dianggap komputer. Kecerdasan diciptakan dan dimasukkan ke dalam suatu mesin (komputer) agar dapat melakukan pekerjaan seperti yang dapat dilakukan manusia. Kemungkinan kecerdasan buatan akan digunakan di masa depan [3]. Salah satu bidang yang bisa diterapkan kecerdasan buatan ialah dunia permainan dengan tujuan untuk memberi tantangan [4].

2.4 *Pathfinding*

Pathfinding adalah sebuah proses pencarian *path* tercepat yang didapat dari titik asal ke titik tujuan dengan menghindari berbagai halangan sepanjang path yang ditempuh [5]. *Pathfinding* memiliki beberapa algoritma yang bisa diterapkan antara lain

Algoritma Dijkstra, Algoritma BFS (*Breadth-First Search*), Algoritma DFS (*Deep-First Search*), Algoritma A* dan beberapa algoritma lainnya.

2.5 Algoritma Dijkstra

Algoritma Dijkstra dinamai menurut penemunya, seorang ilmuwan komputer, Edsger Dijkstra, adalah sebuah algoritma rakus (*greedy algorithm*) yang dipakai dalam memecahkan permasalahan jarak terpendek (*shortest path problem*) untuk sebuah graf berarah (*directed graph*) dengan bobot-bobot sisi (*edge weights*) yang bernilai tidak negatif [6]. Algoritma ini akan mencari jalur dengan *cost minimum*, antara *node* tersebut dengan *node* lainnya. Algoritma ini juga dapat digunakan untuk mencari total biaya (*cost*) dari lintasan terpendek yang dibentuk dari sebuah *node* ke sebuah *node* tujuan.

Berikut adalah tahapan proses perhitungan Algoritma Dijkstra :

1. Menandai semua *node* yang belum pernah dikunjungi dan dimasukkan ke dalam satu *list* yang selanjutnya akan disebut *unvisited*.
2. Menentukan nilai sementara untuk semua *node*, nilai 0 untuk node pertama dan nilai tak hingga untuk yang lainnya. Menandai *node* pertama sebagai *current*.
3. Pada *current node*, semua *node* tetangga yang belum pernah dikunjungi akan dihitung jarak sementaranya. Jarak yang telah dihitung akan dibandingkan dan diambil jarak terpendek.
4. Setelah mempertimbangkan semua *node* tetangga dari *current node*, maka *current* akan ditandai telah dikunjungi dan dihapuskan dari *list unvisited*. *Node* yang telah dikunjungi tidak akan dihitung lagi.
5. Jika *node* destinasi telah dicapai dan ditandai telah dikunjungi maka algoritma telah selesai. Sebaliknya, *node* yang belum dikunjungi dan memiliki nilai sementara

terkecil akan ditandai sebagai *current node* yang baru dan kembali ke tahap 3 [6].

2.6 Unity (Game Engine)

Unity adalah sebuah game engine cross-platform yang dikembangkan oleh Unity Technologies, yang berfungsi untuk mengembangkan sebuah game 3D/2D, dalam berbagai genre, dan untuk berbagai platform dengan sangat mudah dan cepat. Sampai saat ini Unity telah mendukung kurang lebih 27 platform berbeda mulai dari desktop, Web, Android, IOS, XBOX, Nintendo Wii, PS4, dan lainnya. Keunggulan Unity terletak dari mekanisme development game yang jauh lebih ringkas jika dibandingkan dengan menggunakan game engine lainnya [7].

2.7 Bahasa Pemrograman C#

Bahasa C# adalah sebuah bahasa pemrograman modern yang bersifat general-purpose, berorientasi objek, yang dapat digunakan untuk membuat program di atas arsitektur Microsoft .NET Framework. Bahasa C# ini memiliki kemiripan dengan bahasa Java, C dan C++ (selengkapnya dapat dilihat pada Sejarah Bahasa C#).

Bahasa pemrograman ini dikembangkan oleh sebuah tim pengembang di Microsoft yang dipimpin oleh Anders Hejlsberg, seorang yang telah lama malang melintang di dunia pengembangan bahasa pemrograman karena memang ialah yang membuat Borland Turbo Pascal, Borland Delphi, dan juga Microsoft J++.

Kini, C# telah distandarisasi oleh European Computer Manufacturer Association (ECMA) dan juga International Organization for Standardization (ISO) dan telah menginjak versi 3.0 yang mendukung beberapa fitur baru semacam Language Integrated Query (LINQ) dan lain-lainnya [8].

2.8 Mixamo.com

Mixamo.com adalah sebuah satu produk dari Adobe dan layanan animasi karakter *online* pertama yang memiliki beberapa fitur seperti *characters*, *animations*, dan *auto-rigging*. Mixamo.com menyediakan berbagai macam karakter dan animasi yang dapat digunakan. Proses pembuatan animasi di Maximo.com juga sangat cepat dan mudah.

[Halaman ini sengaja dikosongkan]

BAB III
ANALISIS DAN PERANCANGAN SISTEM

Pada tugas akhir ini, akan difokuskan pada *Artificial Intelligence* dengan menggunakan algoritma Dijkstra yang ada pada *game* 'AVIAR'. Bab ini akan membahas mengenai analisis dan perancangan *Artificial Intelligence* pada game 'AVIAR' yang menggunakan algoritma Dijkstra. Pembahasan yang dilakukan meliputi analisis fitur dan perancangan perangkat lunak.

3.1 Analisis Sistem

Pada sub bab ini akan menjelaskan tentang analisis kebutuhan sistem, meliputi spesifikasi kebutuhan sistem, baik itu kebutuhan fungsional sistem maupun kebutuhan non-fungsional sistem, dan identifikasi pengguna sistem.

3.1.1 Spesifikasi Kebutuhan Sistem

Pada sistem ini terdapat beberapa kebutuhan fungsional dan kebutuhan non-fungsional yang mendukung berjalannya sistem. Kebutuhan fungsional sistem dapat dilihat pada **Tabel 3.1** dan kebutuhan non-fungsional sistem dapat dilihat pada **Tabel 3.2**.

Tabel 3.1 Kebutuhan Fungsional Sistem

Kode	Deskripsi
F1	Musuh dapat muncul dalam *maze*
F2	Musuh dapat mengetahui kondisi *node* pada *maze*
F3	Musuh dapat mengetahui posisi pemain
F4	Musuh dapat mencari jalan tercepat menuju pemain
F5	Musuh dapat menyerang pemain

Tabel 3.2 Kebutuhan Non-Fungsional Sistem

Kode	Deskripsi
NF1	Sistem dapat dijalankan di Windows 10
NF2	Sistem memiliki antar muka yang mudah dipahami

3.2 Perancangan Sistem

Sub bab ini akan membahas tentang bagaimana sistem ini dirancang, meliputi deskripsi umum sistem dan arsitektur sistem.

3.2.1 Deskripsi Umum Sistem

AVIAR merupakan *game virtual reality* berbasis personal computer bergenre *puzzle game* yang memanfaatkan *Oculus Rift* sebagai visual pemain dan *Oculus Touch* sebagai kendali pemain. Pembangunan sistem ini dimulai dari membuat *maze* menggunakan *maze generator* yang akan dijadikan sebagai lingkungan nyata di dalam *game* 'AVIAR'.

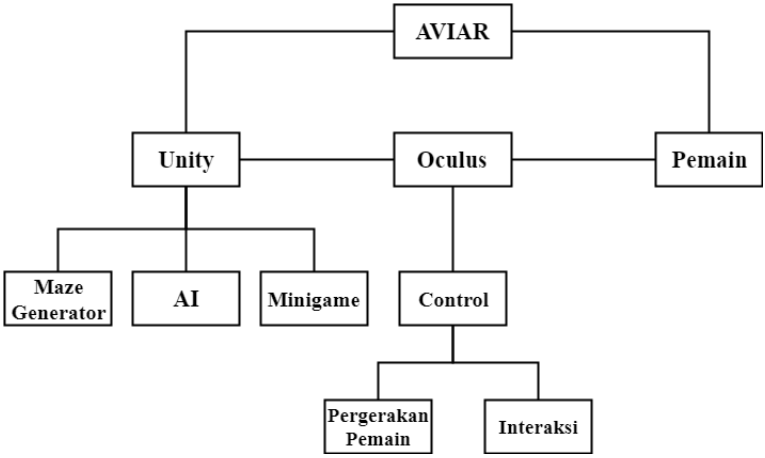
Proses pembangunan sistem selanjutnya yaitu dengan merancang *gameplay* yang sesuai dengan tema dari *game* ini. Mencari *assets* yang sesuai dan melengkapi *assets* dengan *manual*. Terdapat *artificial intelligence* dalam *game* 'AVIAR'.

Artificial Intelligence yang diimplementasikan adalah *pathfinding* dengan menggunakan algoritma Dijkstra yang bertujuan untuk mencari jalan tercepat dari musuh menuju pemain. *Artificial Intelligence* akan dibagi menjadi empat tingkat kesulitan *game* yaitu *easy*, *medium*, *hard*, dan *insane*. Dimana musuh akan memiliki kesempatan yang berbeda-beda yaitu *easy* 50%, *medium* 30%, *hard* 15%, dan *insane* 0%.

3.2.2 Arsitektur Sistem

AVIAR merupakan aplikasi permainan *virtual reality*. AVIAR dibangun menggunakan *tools* Unity3D. Unity3D digunakan untuk membangun *maze generator* yang dikerjakan oleh Anggit Yudhistira, *minigame* yang dikerjakan oleh Aufar Rizqi dan *artificial intelligence* yang dikerjakan oleh Vinsensia Sipriana Zega. Untuk memainkan *game* AVIAR ini harus menggunakan Oculus Touch sebagai kontrol pemain. Kontrol pemain dibagi menjadi dua, yang pertama pergerakan karakter pemain dikerjakan oleh Anggit Yudhistira dan interaksi yang dikerjakan oleh Aufar Rizqi.

Berdasarkan penjelasan di atas, arsitektur sistem dari AVIAR secara sederhana dapat diilustrasikan seperti pada **Gambar 3.1**.



Gambar 3.1 Arsitektur Sistem

3.3 Penerapan *Artificial Intelligence*

Penerapan *Artificial Intelligence* pada musuh di *game* AVIAR dibuat dengan bantuan algoritma Dijkstra sebagai algoritma *pathfinding*. Algoritma tersebut dikombinasikan dengan *game* AVIAR ini. Pada bagian ini akan dibagi menjadi dua sub-bagian, yaitu penerapan *Artificial Intelligence* menggunakan algoritma Dijkstra, dan penerapan *Artificial Intelligence* pada tingkat kesulitan *game*.

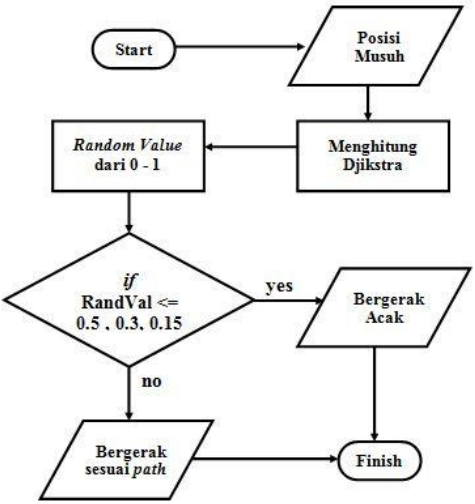
3.3.1 Penerapan *Node* sebagai *Path* menggunakan Model *Maze*

Model *maze* yang akan digunakan dibangun oleh Anggit Yudhistira menggunakan algoritma *Recursive Division*. Model *maze* yang telah didapat akan dibentuk kembali menjadi objek *node* yang memiliki nilai 0 dan 1, dimana yang bernilai 0 tidak dapat dilewati dan sebaliknya yang bernilai 1 dapat dilewati.

3.3.2 Penerapan *Artificial Intelligence* menggunakan Algoritma Dijkstra

Penerapan *Artificial Intelligence* dibuat dengan menggabungkan algoritma Dijkstra *pathfinding* dengan aturan *game*.

1. *Nodes*
   Nodes merupakan kumpulan poin yang saling berhubungan satu sama lain. Setiap *node* memiliki informasi yang diperlukan untuk pergerakan musuh di dalam labirin. Bobot atau *cost* antar *node* ialah satu, karena dalam permainan *maze* jarak antar *node* dianggap sama.
2. Penerapan algoritma Dijkstra *pathfinding*
   Untuk *multiplayer*, musuh akan menghitung jarak dari kedua pemain yang akan dibandingkan dan dipilih sebagai *shortest path*. Diagram alir algoritma Dijkstra *Pathfinding* dapat dilihat pada **Gambar 3.2**. *Random value* dari 0 sampai 1 digunakan untuk menjadi pemicu pergerakan acak yang berbeda-beda pada setiap tingkat kesulitan. *Random value* pada setiap tingkat kesulitan dapat dilihat pada **Tabel 3.3**.

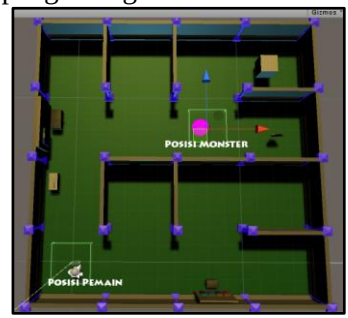


Gambar 3.2 Diagram Alir Algoritma Dijkstra *Pathfinding*

Penerapan algoritma Dijkstra pada *artificial intelligence* dapat ditunjukkan melalui penjelasan berikut.

a. Kondisi awal

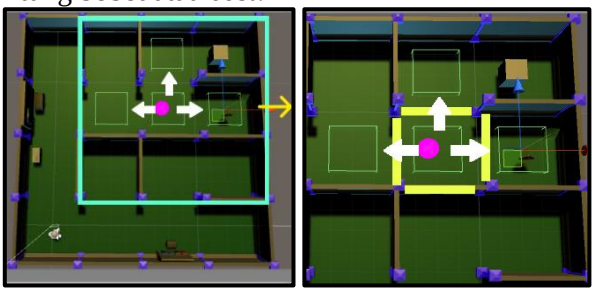
Pada **Gambar 3.3**, terlihat kondisi awal dalam *maze* terdapat posisi pemain, posisi musuh dan dinding *maze* yang menjadi penghalang.



Gambar 3.3 Kondisi awal pada *maze*

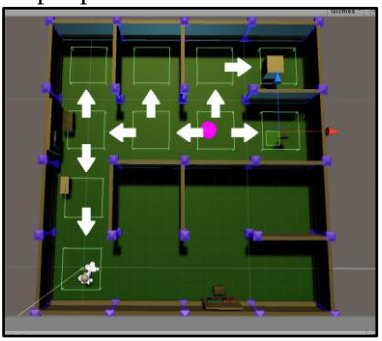
b. Langkah awal

Pada **Gambar 3.4**, *artificial intelligence* akan mencoba seluruh *node* yang bertetangga dengan posisi awal. *Node* tetangga yang dimaksud ialah *node* yang bersebelahan dengan posisi awal. *Node* yang bertetangga ditandai dengan garis berwarna kuning. Untuk setiap pergerakan akan dihitung bobot atau *cost*.



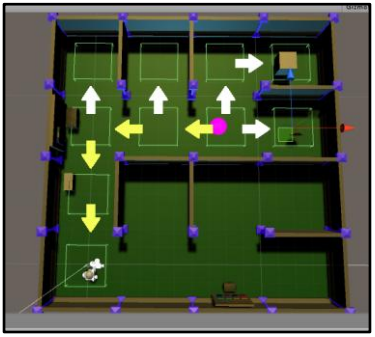
Gambar 3.4 Langkah awal

c. *Trace* dari *initial state* sampai *goal state*
Setiap *node* yang dianggap sebagai kemungkinan *path* akan diberi nilai berdasarkan fungsi evaluasi dengan cara melakukan *looping* untuk mencoba semua kemungkinan. Kemungkinan *path* ditandai dengan tanda panah berwarna putih yang terdapat pada **Gambar 3.5**.



Gambar 3.5 *Trace Initial State* ke *Goal State*

d. Memilih *solution path*
Setelah melakukan *trace initial state* ke *goal state*, *artificial intelligence* akan menemukan *solution path* dengan *cost* terendah. *Solution path* ditandai dengan tanda panah berwarna kuning yang terdapat pada **Gambar 3.6**.



Gambar 3.6 *Solution Path*

3.3.3 Penerapan *Artificial Intelligence* pada tingkat kesulitan *game*

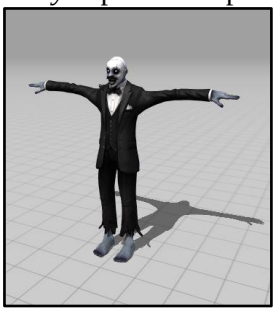
Artificial Intelligence pada tingkat kesulitan *game* dibagi menjadi empat bagian yang bisa dilihat pada **Tabel 3.3**. Setiap tingkat kesulitan *game* memiliki karakter musuh yang berbeda-beda. *Asset* musuh untuk setiap tingkat kesulitan *game* diambil dari situs Mixamo.com.

Tabel 3.3 Pembagian tingkat kesulitan *game*

Tingkat Kesulitan	Batasan Level	Persentase Acak
Easy	Level 1 – 4	50%
Medium	Level 5 – 8	30%
Hard	Level 9 – 12	15%
Insane	Level 13 – 16	0%

3.3.3.1 Penerapan Artificial Intelligence pada tingkat kesulitan Easy

Artificial intelligence pada tingkat kesulitan *easy* akan memiliki kemungkinan untuk bergerak secara acak sebesar 50%. Gerakan acak yang dimaksud adalah ketika musuh telah melakukan pencarian jalan tercepat menggunakan algoritma Dijkstra dan menemukan jalan yang dituju, musuh tersebut akan memiliki kemungkinan untuk bergerak tidak sesuai dengan solusi optimal yang telah didapatkan. *Asset* musuh yang akan muncul pada tingkat kesulitan *easy* dapat dilihat pada **Gambar 3.7**.



Gambar 3.7 Musuh pada tingkat kesulitan *easy*

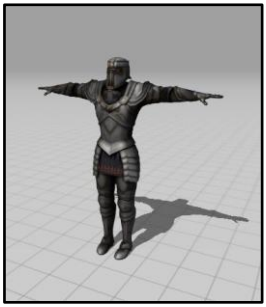
3.3.3.2 Penerapan Artificial Intelligence pada tingkat kesulitan Medium

Artificial intelligence pada tingkat kesulitan *medium* akan memiliki kemungkinan untuk bergerak secara acak sebesar 30%. Gerakan acak yang dimaksud adalah ketika musuh telah melakukan pencarian jalan tercepat menggunakan algoritma Dijkstra dan menemukan jalan yang dituju, musuh tersebut akan memiliki kemungkinan untuk bergerak tidak sesuai dengan solusi optimal yang telah didapatkan. *Asset* musuh yang akan muncul pada tingkat kesulitan *medium* dapat dilihat pada **Gambar 3.8**.

Gambar 3.8 Musuh pada tingkat kesulitan *medium*

3.3.3.3 Penerapan Artificial Intelligence pada tingkat kesulitan Hard

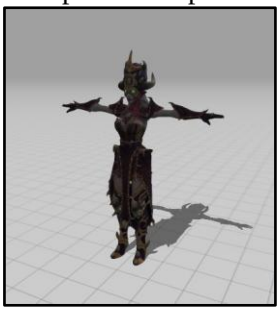
Artificial intelligence pada tingkat kesulitan *hard* akan memiliki kemungkinan untuk bergerak secara acak sebesar 15%. Gerakan acak yang dimaksud adalah ketika musuh telah melakukan pencarian jalan tercepat menggunakan algoritma Dijkstra dan menemukan jalan yang dituju, musuh tersebut akan memiliki kemungkinan untuk bergerak tidak sesuai dengan solusi optimal yang telah didapatkan. *Asset* musuh yang akan muncul pada tingkat kesulitan *hard* dapat dilihat pada **Gambar 3.10**.



Gambar 3.9 Musuh pada tingkat kesulitan *hard*

3.3.3.4 Penerapan Artificial Intelligence pada tingkat kesulitan Insane

Artificial intelligence pada tingkat kesulitan *insane* tidak memiliki pergerakan acak dengan kata lain pergerakan yang dilakukan akan mengikuti solusi optimal yang telah didapatkan oleh algoritma Dijkstra. *Asset* musuh yang akan muncul pada tingkat kesulitan *insane* dapat dilihat pada **Gambar 3.10**.



Gambar 3.10 Musuh pada tingkat kesulitan *insane*

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

Bab ini membahas implementasi yang dilakukan berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya. Sebelum penjelasan implementasi akan ditunjukkan terlebih dahulu lingkungan untuk melakukan implementasi.

Pada bagian implementasi ini juga akan dijelaskan mengenai fungsi-fungsi yang digunakan dalam program tugas akhir ini dan disertai dengan kode sumber masing-masing fungsi utama.

4.1 Lingkungan Implementasi

Spesifikasi perangkat keras serta perangkat lunak yang digunakan dalam tahap implementasi perangkat lunak tugas akhir ini dijelaskan pada **Tabel 4.1**.

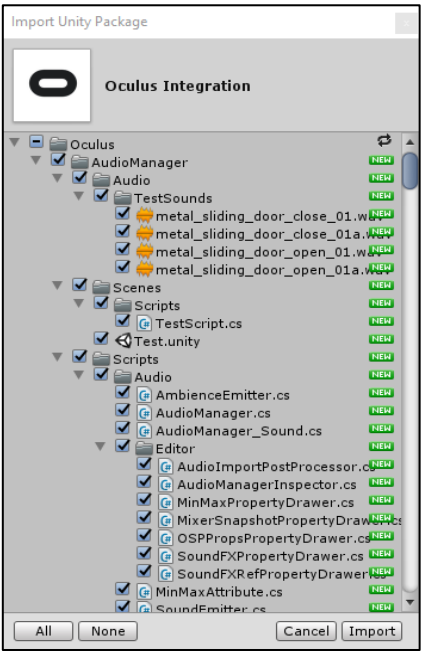
Tabel 4.1 Lingkungan Implementasi

Perangkat	Spesifikasi
Perangkat Keras	✓ Prosesor Intel(R) Core (TM) i7-770 CPU @ 3.60GHz (8 CPUs), ~3.6 GHz
	✓ RAM 8 Gigabyte
	✓ Graphic card NVIDIA GeForce GTX 1060 3GB
	✓ Oculus Rift
Perangkat Lunak	✓ Sistem operasi Windows 10 Home Single Language 64
	✓ Unity 2017.3.0f3 Personal (64-bit)

4.2 Implementasi *Import Asset* dan *Package*

Game ‘AVIAR’ ini berbasis *Virtual Reality* dan menggunakan perangkat keras Oculus Rift, jadi penulis

menggunakan *package* Oculus Rift yang sudah *compatible* dengan Unity yang bernama "Oculus Integration". *Package* tersebut diunduh melalui *Asset Store* dari Unity langsung secara gratis. Caranya dengan memilih *tab window* pada Unity dan pilih *asset store*, kemudian masukkan kata kunci "Oculus" pada kolom *search* dan pilih "Oculus Integration" kemudian klik tombol *download.* Tunggu sampai selesai kemudian klik tombol *import*. Kemudian akan keluar *window* seperti **Gambar 4.1**, kemudian klik tombol *import* dan tunggu hingga proses *load asset* selesai. Apabila *load asset* sudah selesai, *Asset* berhasil di*import* ke dalam *project.*



Gambar 4.1 Implementasi *Import Asset*

4.3 Implementasi Algoritma Djikstra *Pathfinding*

Dalam menemukan pemain, musuh akan bergerak menggunakan algoritma Dijkstra. Algoritma Dijkstra *pathfinding* dapat dilihat pada **Kode Sumber 4.1.**

```
1.  using System.Collections;
2.  using System.Collections.Generic;
3.  using UnityEngine;
4.
5.  public class Dijkstra : MonoBehaviour {
6.
7.      private GameObject[] nodes;
8.
9.      public List<Transform> findShortestPath(Transform
     start, Transform end) {
10.         nodes = GameObject.FindGameObjectsWithTag("No
    de");
11.
12.         List<Transform> result = new List<Transform>(
    );
13.         result.Clear();
14.         Transform node = DijkstraAlgo(start, end);
15.
16.         while (node != null) {
17.             result.Add(node);
18.             Node currentNode = node.GetComponent<Node
    >();
19.             node = currentNode.getParentNode();
20.         }
21.
22.         return result;
23.     }
24.
25.     private Transform DijkstraAlgo(Transform start, T
    ransform end) {
26.         List<Transform> unexplored = new List<Transfo
    rm>();
27.
28.         // mendata nodes yang belum dikunjungi
29.         foreach (GameObject obj in nodes) {
30.             Node n = obj.GetComponent<Node>();
31.             if (n.isWalkable()) {
32.                 n.resetNode();
33.                 unexplored.Add(obj.transform);
34.             }
35.         }
```

```
36.          Node startNode = start.GetComponent<Node>();
37.          startNode.setWeight(0);
38.
39.          // mengurutkan nodes berdasarkan bobot terkecil
40.          while (unexplored.Count > 0) {
41.              unexplored.Sort((x, y) => x.GetComponent<Node>().getWeight().CompareTo(y.GetComponent<Node>().getWeight()));
42.
43.              Transform current = unexplored[0];
44.
45.              if (current == end)
46.                  return end;
47.
48.              // menghapus node yang telah dikunjungi
49.              unexplored.Remove(current);
50.
51.              Node currentNode = current.GetComponent<Node>();
52.              // memanggil fungsi untuk mendapatkan node tetagga
53.              List<Transform> neighbors = currentNode.getNeighbourNode();
54.
55.              // menghitung jarak ke semua node tetangga yang dapat dilewati dan belum pernah dikunjungi
56.              foreach (Transform neighNode in neighbors) {
57.                  Node node = neighNode.GetComponent<Node>();
58.
59.                  if (unexplored.Contains(neighNode) && node.isWalkable()) {
60.                      float distance = Vector3.Distance(neighNode.position, current.position);
61.                      distance = currentNode.getWeight() + distance;
62.
63.                      // node tetangga dengan jarak terkecil akan dijadikan sebagai node aktif selanjutnya dan node aktif saat ini akan diset sebagai parent node
64.                      if (distance < node.getWeight())
{
```

```
65.                             node.setWeight(distance);
66.                             node.setParentNode(current);

67.                         }
68.                     }
69.                 }
70.             }
71.             return end;
72.         }
73. }
```

Kode Sumber 4.1 Implementasi Algoritma Dijkstra *Pathfinding*

Algoritma Dijkstra diimplementasi dengan menggunakan *priority queue* yaitu pasangan dari setiap *neighbour* dan *cost* yang harus dikeluarkan untuk mencapai kesana, dimana akan menghitung seluruh kemungkinan jalan terdekat untuk mencapai tujuan.

4.4 Implementasi pada Musuh

Pada sub bab ini akan membahas implementasi dari musuh yang dituliskan dengan menggunakan bahasa pemrograman C#.

4.4.1 Implementasi *health* pada Musuh

Musuh memiliki nilai *health* yang dapat berkurang apabila diserang oleh pemain menggunakan senjata. Apabila nilai *health* musuh mencapai nol, musuh akan mati. Implementasi *health* pada musuh dapat dilihat pada **Kode Sumber 4.2**.

```
1.  private void OnTriggerEnter(Collider other)
2.      {
3.          if (other.gameObject.tag == "Player") {
4.              a1.Play();
5.          }
6.          if (other.gameObject.tag == "weapon01" && flaghit
    == false) {
7.              hit.Play();
8.              this.monshp -= 5;
9.              flaghit = true;
10.
```

```
        }
        if (other.gameObject.tag == "weapon02" && flaghit
    == false) {
            hit.Play();
            this.monshp -= 10;
            flaghit = true;
        }
        if (other.gameObject.tag == "weapon03" && flaghit
    == false)  {
            hit.Play();
            this.monshp -= 15;
            flaghit = true;

        }
        if (this.monshp <= 0) {
            Destroy(this.gameObject);
        }
    }
```

Kode Sumber 4.2 Implementasi *health* pada musuh

4.4.2 Implementasi *movement* pada musuh

Musuh dapat bergerak mencari pemain saat berada dalam *maze*. Pergerakkan dari musuh akan berbeda berdasarkan tingkat kesulitan *game*. Perbedaan tersebut ditentukan dengan membedakan tingkat ketepatan musuh bergerak ke arah pemain. Implementasi *movement* pada musuh dapat dilihat pada **Kode Sumber 4.3**.

```
void Update () {
        startNodeP1 = playerComponent[0].startNode;
        if (playerPrefab.Length > 1) {
            startNodeP2 = playerComponent[1].startNode;
        }

        if (playerPrefab.Length > 1) {
            if (P1 == true) {

                // mendapatkan arah dari musuh ke pemain
                direction = playerComponent[0].transform.p
    osition - this.transform.position;
```

```
12.                 direction.y = 0;
13.                 // mendapatkan jarak dari musuh ke pemain
14.                 distance = Vector3.Distance(playerComponen
    t[0].transform.position, this.transform.position);
15.             }
16.             else if(P2 == true) {
17.                 direction = playerComponent[1].transform.p
    osition - this.transform.position;
18.                 direction.y = 0;
19.                 distance = Vector3.Distance(playerComponen
    t[1].transform.position, this.transform.position);
20.             }
21.         }
22.         else {
23.             direction = playerComponent[0].transform.posit
    ion - this.transform.position;
24.             direction.y = 0;
25.             distance = Vector3.Distance(playerComponent[0]
    .transform.position, this.transform.position);
26.         }
27.
28.         // jika pemain berada dalam jangkauan musuh maka
    musuh akan menyerang pemain
29.         if (distance <= 1.5f) {
30.             this.transform.rotation = Quaternion.Slerp(thi
    s.transform.rotation, Quaternion.LookRotation(direction),
    0.75f);
31.             anim.SetBool("IsAttacking", true);
32.             anim.SetBool("IsRunning", false);
33.             anim.SetBool("IsIdling", false);
34.         }
35.         // jika pemain tidak berada dalam jangkauan musuh
    maka musuh berjalan mengikuti path yang telah dihitung
    menggunakan algoritma Dijkstra
36.         else if (distance > 1.5f) {
37.             GameObject current = paths[0];
38.             Vector3 directionNode = current.transform.posi
    tion - this.transform.position;
39.             directionNode.y = 0;
40.
41.             this.transform.rotation = Quaternion.Slerp(thi
    s.transform.rotation, Quaternion.LookRotation(directionNod
    e), 0.75f);
```

```
42.             this.transform.position = Vector3.MoveTowards(
    this.transform.position, current.transform.position, step)
    ;
43.             anim.SetBool("IsRunning", true);
44.             anim.SetBool("IsIdling", false);
45.             anim.SetBool("IsAttacking", false);
46.         }
47.     }
```

Kode Sumber 4.3 Implementasi *movement* pada Musuh

4.4.2.1 Implementasi musuh pada tingkat kesulitan easy

Pada tingkat kesulitan *easy*, musuh akan bergerak secara random sebesar 50% dalam menemukan *path* yang dituju. Implementasi musuh pada tingkat kesulitan *easy* dapat dilihat pada **Kode Sumber 4.4**.

```
1.  private void OnTriggerEnter(Collider other)
2.      {
3.          if (other.gameObject.tag == "Node") {
4.              endNode = other.gameObject;
5.              pathsP1 = finder.findShortestPath(startNodeP1,
     endNode);
6.              P1 = false;
7.              if (playerPrefab.Length > 1) {
8.                  pathsP2 = finder.findShortestPath(startNod
    eP2, endNode);
9.                  P2 = false;
10.
11.                 if (pathsP1.Count >= pathsP2.Count) {
12.                     paths = pathsP1;
13.                     P1 = true;
14.                 } else {
15.                     paths = pathsP2;
16.                     P2 = true;
17.                 }
18.             }
19.             else
20.                 paths = pathsP1;
21.
```

```
22.
23.             GameObject current = paths[0];
24.             random = Random.Range(0.0f, 1.0f);
25.             // untuk tingkat kesulitan easy, musuh
    memiliki kesempatan untuk bergerak secara acak sebesar 50%
26.             if (random <= 0.5f) {
27.                 Node node = current.GetComponent<Node>();

28.                 List<GameObject> neighbors = node.getNeigh
    bourNode();
29.                 bool flag = false;
30.                 // jika musuh bergerak secara acak, akan
    dilakukan pengecekan node tetangga yang dapat dilewati
    secara acak
31.                 if (neighbors.Count > 1) {
32.                     while (flag != true) {
33.                         int next = Random.Range(0, neighbo
    rs.Count);
34.                         Node nextNode = neighbors[next].Ge
    tComponent<Node>();
35.                         if (nextNode.isWalkable()) {
36.                             paths[1] = nextNode.gameObject
    ;
37.                             paths.Remove(current);
38.                             flag = true;
39.                             break;
40.                         }
41.                     }
42.                 }
43.             }
44.             else {
45.                 paths.Remove(current);
46.                 current = paths[0];
47.             }
48.         }
49.     }
```

Kode Sumber 4.4 Implementasi musuh pada tingkat kesulitan *easy*

4.4.2.2 Implementasi musuh pada tingkat kesulitan medium

Pada tingkat kesulitan *medium*, musuh akan bergerak secara random sebesar 30% dalam menemukan *path* yang dituju. Implementasi musuh pada tingkat kesulitan *medium* dapat dilihat pada **Kode Sumber 4.5**.

```
1.  private void OnTriggerEnter(Collider other)
2.      {
3.          if (other.gameObject.tag == "Node") {
4.              endNode = other.gameObject;
5.              pathsP1 = finder.findShortestPath(startNodeP1,
     endNode);
6.              P1 = false;
7.              if (playerPrefab.Length > 1) {
8.                  pathsP2 = finder.findShortestPath(startNod
    eP2, endNode);
9.                  P2 = false;
10.
11.                 if (pathsP1.Count >= pathsP2.Count) {
12.                     paths = pathsP1;
13.                     P1 = true;
14.                 } else {
15.                     paths = pathsP2;
16.                     P2 = true;
17.                 }
18.             }
19.             else
20.                 paths = pathsP1;
21.
22.
23.             GameObject current = paths[0];
24.             random = Random.Range(0.0f, 1.0f);
25.             // untuk tingkat kesulitan medium, musuh
    memiliki kesempatan untuk bergerak secara acak sebesar 30%
26.             if (random <= 0.3f) {
27.                 Node node = current.GetComponent<Node>();

28.                 List<GameObject> neighbors = node.getNeigh
    bourNode();
29.
30.                 bool flag = false;
31.                 // jika musuh bergerak secara acak, akan
    dilakukan pengecekan node tetangga yang dapat dilewati
```

```
    secara acak
32.                 if (neighbors.Count > 1) {
33.                     while (flag != true) {
34.                         int next = Random.Range(0, neighbo
    rs.Count);
35.                         Node nextNode = neighbors[next].Ge
    tComponent<Node>();
36.                         if (nextNode.isWalkable()) {
37.                             paths[1] = nextNode.gameObject
    ;
38.                             paths.Remove(current);
39.                             flag = true;
40.                             break;
41.                         }
42.                     }
43.                 }
44.             }
45.             else {
46.                 paths.Remove(current);
47.                 current = paths[0];
48.             }
49.         }
50.     }
```

Kode Sumber 4.5 Implementasi musuh pada tingkat kesulitan *medium*

4.4.2.3 Implementasi musuh pada tingkat kesulitan hard

Pada tingkat kesulitan *hard*, musuh akan bergerak secara random sebesar 15% dalam menemukan *path* yang dituju. Implementasi musuh pada tingkat kesulitan *hard* dapat dilihat pada **Kode Sumber 4.6**.

```
1.  private void OnTriggerEnter(Collider other)
2.      {
3.          if (other.gameObject.tag == "Node") {
4.              endNode = other.gameObject;
5.              pathsP1 = finder.findShortestPath(startNodeP1,
     endNode);
6.              P1 = false;
```

```
7.              if (playerPrefab.Length > 1) {
8.                  pathsP2 = finder.findShortestPath(startNod
    eP2, endNode);
9.                  P2 = false;
10.
11.                 if (pathsP1.Count >= pathsP2.Count) {
12.                     paths = pathsP1;
13.                     P1 = true;
14.                 } else {
15.                     paths = pathsP2;
16.                     P2 = true;
17.                 }
18.             }
19.             else
20.                 paths = pathsP1;
21.
22.
23.             GameObject current = paths[0];
24.             random = Random.Range(0.0f, 1.0f);
25.             // untuk tingkat kesulitan hard, musuh
    memiliki kesempatan untuk bergerak secara acak sebesar 15%
26.             if (random <= 0.15f) {
27.                 Node node = current.GetComponent<Node>();

28.                 List<GameObject> neighbors = node.getNeigh
    bourNode();
29.
30.                 bool flag = false;
31.                 // jika musuh bergerak secara acak, akan
    dilakukan pengecekan node tetangga yang dapat dilewati
    secara acak
32.                 if (neighbors.Count > 1) {
33.                     while (flag != true) {
34.                         int next = Random.Range(0, neighbo
    rs.Count);
35.                         Node nextNode = neighbors[next].Ge
    tComponent<Node>();
36.                         if (nextNode.isWalkable()) {
37.                             paths[1] = nextNode.gameObject
    ;
38.                             paths.Remove(current);
39.                             flag = true;
40.                             break;
```

```
41.                         }
42.                     }
43.                 }
44.             }
45.             else {
46.                 paths.Remove(current);
47.                 current = paths[0];
48.             }
49.         }
50.     }
```

Kode Sumber 4.6 Implementasi musuh pada tingkat kesulitan *hard*

4.4.2.4 Implementasi musuh pada tingkat kesulitan insane

Pada tingkat kesulitan *insane*, musuh tidak akan bergerak secara random dalam menemukan *path* yang dituju. Implementasi musuh pada tingkat kesulitan *insane* dapat dilihat pada **Kode Sumber 4.7**.

```
1.  private void OnTriggerEnter(Collider other)
2.      {
3.          if (other.gameObject.tag == "Node") {
4.              endNode = other.gameObject;
5.              pathsP1 = finder.findShortestPath(startNodeP1,
     endNode);
6.              P1 = false;
7.              if (playerPrefab.Length > 1) {
8.                  pathsP2 = finder.findShortestPath(startNod
    eP2, endNode);
9.                  P2 = false;
10.
11.                 if (pathsP1.Count >= pathsP2.Count) {
12.                     paths = pathsP1;
13.                     P1 = true;
14.                 } else {
15.                     paths = pathsP2;
16.                     P2 = true;
17.                 }
18.             }
19.             else
```

```
20.                 paths = pathsP1;
21.
22.             GameObject current = paths[0];
23.             paths.Remove(current);
24.             current = paths[0];
25.         }
26.     }
```

Kode Sumber 4.7 Implementasi musuh pada tingkat kesulitan *insane*

BAB V
UJI COBA DAN EVALUASI

Pada bab ini akan dijelaskan mengenai rangkaian uji coba dan evaluasi yang dilakukan. Proses pengujian dilakukan menggunakan metode *black box* berdasarkan skenario yang telah ditentukan.

5.1 Lingkungan Uji Coba

Pada proses pengujian perangkat lunak, dibutuhkan suatu lingkungan pengujian yang sesuai dengan standar kebutuhan. Lingkungan pengujian sistem pada pengerjaan Tugas Akhir ini dilakukan pada setiap tingkat kesulitan kemampuan *Artificial Intelligence*. Spesifikasi lingkungan pengujian dijabarkan pada **Tabel 5.1**.

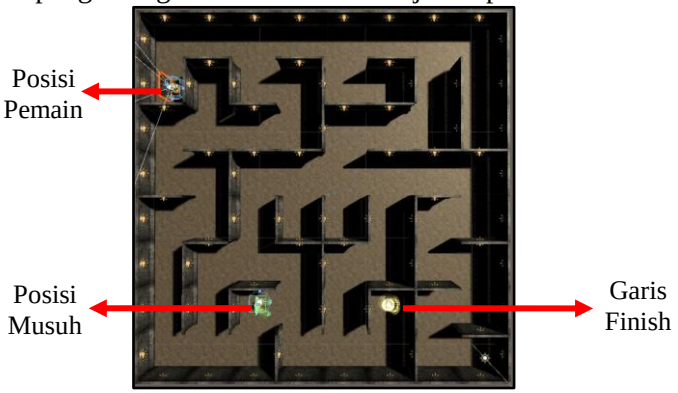
Tabel 5.1 Lingkungan Pengujian Sistem

Perangkat	Spesifikasi
Perangkat Keras	✓ Prosesor Intel(R) Core (TM) i7-770 CPU @ 3.60GHz (8 CPUs), ~3.6 GHz
	✓ RAM 8 Gigabyte
	✓ Graphic card NVIDIA GeForce GTX 1060 3GB
	✓ Oculus Rift
Perangkat Lunak	✓ Sistem operasi Windows 10 Home Single Language 64
	✓ Unity 2017.3.0f3 Personal (64-bit)

5.2 Pengujian Fungsionalitas

Pengujian fungsionalitas sistem dilakukan dengan menyiapkan sejumlah skenario sebagai tolok ukur keberhasilan pengujian. Pengujian fungsionalitas dilakukan dengan kondisi *maze* sintetis yang berukuran 8x8 dan kondisi awal yang sama

yaitu berupa posisi pemain, posisi musuh dan dinding *maze* yang menjadi penghalang. Kondisi *maze* ditunjukan pada **Gambar 5.1**.



Gambar 5.1 Kondisi awal *maze*

Pengujian fungsionalitas yang terdapat pada *game* dijabarkan sebagai berikut.

5.2.1 Uji Coba Musuh Muncul Dalam *Maze*

Pada sub bab ini dijelaskan secara detail mengenai skenario yang dilakukan dan hasil yang didapatkan dari pengujian fungsionalitas musuh muncul dalam *maze*. Penjelasan disajikan dengan menampilkan kondisi awal, masukan, keluaran, hasil yang dicapai, dan kondisi akhir. Skenario yang telah diuji terdapat pada **Tabel 5.3**.

Tabel 5.2 Hasil Uji Coba Musuh Muncul Dalam Maze

ID	UF-001
Nama	Uji coba musuh muncul dalam *maze*
Tujuan uji coba	Musuh berhasil muncul dalam *maze*
Kondisi Awal	Waktu siang hari dan musuh belum muncul dalam *maze*
Skenario 1	***Musuh muncul dalam maze***
Masukan	Waktu berubah menjadi malam hari

Keluaran yang diharapkan	Musuh muncul dalam *maze*
Hasil Uji	Musuh berhasil muncul dalam *maze* pada malam hari

5.2.2 Uji Coba Musuh Mengetahui Kondisi *Node* pada *Maze*

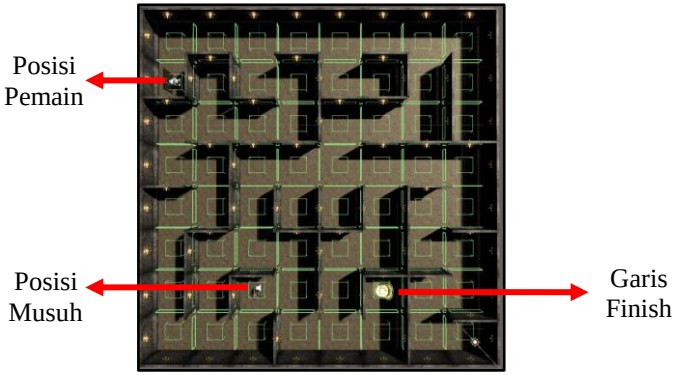
Pada sub bab ini dijelaskan secara detail mengenai skenario yang dilakukan dan hasil yang didapatkan dari pengujian fungsionalitas musuh mengetahui kondisi *node* pada *maze*. Penjelasan disajikan dengan menampilkan kondisi awal, masukan, keluaran, hasil yang dicapai, dan kondisi akhir. Skenario yang telah diuji terdapat pada **Tabel 5.3**.

Tabel 5.3 Hasil Uji Coba Musuh Mengetahui Kondisi *Node* pada *Maze*

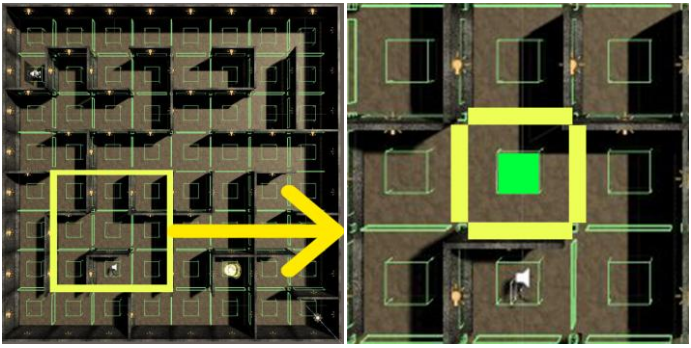
ID	UF-002
Nama	Uji coba musuh mengetahui kondisi *node* pada *maze*
Tujuan uji coba	Musuh mengetahui kondisi *node* pada *maze*
Kondisi Awal	Musuh belum menyentuh *collider node*
Skenario 1	***Musuh pada tingkat kesulitan easy mengetahui kondisi node pada maze***
Masukan	Musuh memasuki *collider node*
Keluaran yang diharapkan	Musuh mengetahui kondisi *node* pada *maze*
Hasil Uji	Musuh pada tingkat kesulitan *easy* berhasil mengetahui kondisi *node* pada *maze*
Skenario 2	***Musuh pada tingkat kesulitan medium mengetahui kondisi node pada maze***
Masukan	Musuh memasuki *collider node*

Keluaran yang diharapkan	Musuh mengetahui kondisi *node* pada *maze*
Hasil Uji	Musuh pada tingkat kesulitan *medium* berhasil mengetahui kondisi *node* pada *maze*
Skenario 3	***Musuh pada tingkat kesulitan hard mengetahui kondisi node pada maze***
Masukan	Musuh memasuki *collider node*
Keluaran yang diharapkan	Musuh mengetahui kondisi *node* pada *maze*
Hasil Uji	Musuh pada tingkat kesulitan *hard* berhasil mengetahui kondisi *node* pada *maze*
Skenario 4	***Musuh pada tingkat kesulitan insane mengetahui kondisi node pada maze***
Masukan	Musuh memasuki *collider node*
Keluaran yang diharapkan	Musuh mengetahui kondisi *node* pada *maze*
Hasil Uji	Musuh pada tingkat kesulitan *insane* berhasil mengetahui kondisi *node* pada *maze*

Hasil uji dari musuh mengetahui kondisi *maze* dapat dilihat pada **Gambar 5.2**. Dalam *maze* terdapat posisi *node* dimana setiap *node* memiliki *neighbour* yang disebut sebagai *adjacency node* yang dapat dilihat pada **Gambar 5.3**, posisi *node* yang sedang aktif ditunjukan oleh kotak berwarna hijau, *neighbour* antar *node* ditunjukan oleh kotak berwarna kuning, dan *node* yang tidak termasuk sebagai *neighbour* ditunjukan oleh kotak berwarna putih.



Gambar 5.2 Hasil Uji Coba Musuh Mengetahui Kondisi *Maze*

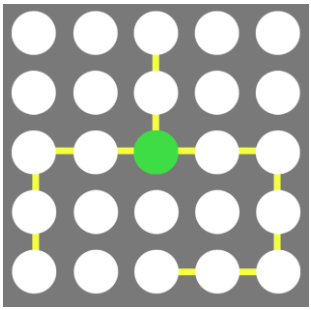


Gambar 5.3 Ilustrasi *neighbour* dari *node*

Dari **Gambar 5.3**, didapatkan matriks *adjacency node* dan graf *adjacency node*. Matriks *adjacency node* dapat dilihat pada **Gambar 5.4**. Matriks yang bernilai 0 menandakan bahwa *node* tersebut tidak dapat dilewati dan sebaliknya, *node* yang bernilai 1 dapat dilewati. Posisi *node* yang sedang aktif ditandai dengan warna hijau dan *neighbour* antar *node* ditandai dengan warna kuning. Graf *adjacency node* dapat dilihat pada **Gambar 5.5**. Posisi *node* yang sedang aktif ditandai dengan warna hijau dan graf yang saling berhubungan ditandai dengan garis warna kuning.

1	0	1	0	1
0	0	1	0	0
1	1	1	1	1
1	0	0	0	1
1	0	1	1	1

Gambar 5.4 Matriks *Adjacency Node*



Gambar 5.5 Graf *Adjacency Node*

5.2.3 Uji Coba Musuh Mengetahui Posisi Pemain

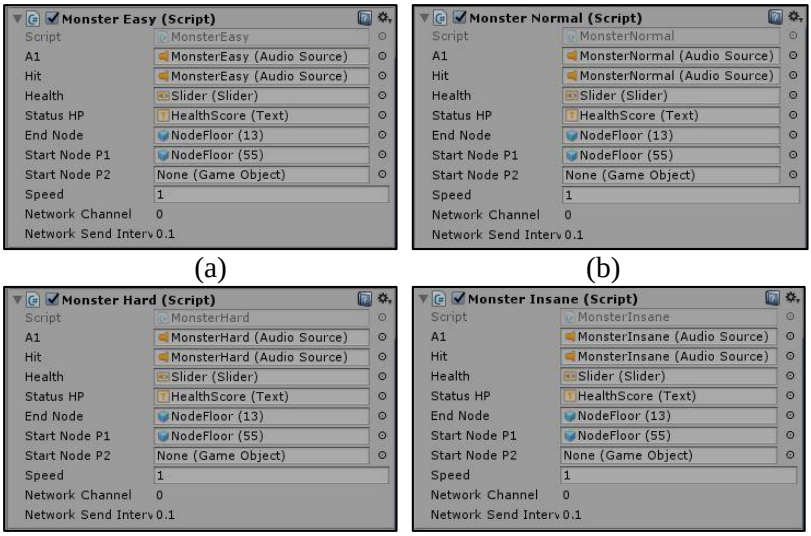
Pada sub bab ini dijelaskan secara detail mengenai skenario yang dilakukan dan hasil yang didapatkan dari pengujian fungsionalitas musuh mengetahui posisi pemain. Penjelasan disajikan dengan menampilkan kondisi awal, masukan, keluaran, hasil yang dicapai, dan kondisi akhir. Skenario yang telah diuji terdapat pada **Tabel 5.4**.

Tabel 5.4 Hasil Uji Coba Musuh Mengetahui Posisi Pemain

ID	UF-003
Nama	Uji coba musuh mengetahui posisi pemain
Tujuan uji coba	Musuh mengetahui posisi dari pemain
Kondisi Awal	Pemain berada pada *maze*
Skenario 1	***Musuh pada maze tingkat kesulitan easy mengetahui posisi pemain***
Masukan	Musuh memasuki *collider node*
Keluaran yang diharapkan	Musuh mengetahui posisi pemain
Hasil Uji	Musuh pada *maze* tingkat kesulitan *easy* berhasil mengetahui posisi pemain
Skenario 2	***Musuh pada maze tingkat kesulitan medium***

	mengetahui posisi pemain
Masukan	Musuh memasuki *collider node*
Keluaran yang diharapkan	Musuh mengetahui posisi pemain
Hasil Uji	Musuh pada *maze* tingkat kesulitan *medium* berhasil mengetahui posisi pemain
Skenario 3	***Musuh pada maze tingkat kesulitan hard mengetahui posisi pemain***
Masukan	Musuh memasuki *collider node*
Keluaran yang diharapkan	Musuh mengetahui posisi pemain
Hasil Uji	Musuh pada *maze* tingkat kesulitan *hard* berhasil mengetahui posisi pemain
Skenario 4	***Musuh pada maze tingkat kesulitan insane mengetahui posisi pemain***
Masukan	Musuh memasuki *collider node*
Keluaran yang diharapkan	Musuh mengetahui posisi pemain
Hasil Uji	Musuh pada *maze* tingkat kesulitan *insane* berhasil mengetahui posisi pemain

Hasil uji dari skenario 1 yaitu musuh mengetahui posisi pemain pada *maze* tingkat kesulitan *easy* yang dapat dilihat pada **Gambar 5.6(a)**. Selanjutnya hasil uji skenario 2 yaitu musuh mengetahui posisi pemain pada maze tingkat kesulitan *medium* yang dapat dilihat pada **Gambar 5.6(b)**. Selanjutnya hasil uji skenario 3 yaitu musuh mengetahui posisi pemain pada maze tingkat kesulitan *hard* yang dapat dilihat pada **Gambar 5.6(c)**. Selanjutnya hasil uji skenario 4 yaitu musuh mengetahui posisi pemain pada maze tingkat kesulitan *insane* dapat dilihat pada **Gambar 5.6(d)**.



Gambar 5.6 Hasil Uji Coba Musuh Mengetahui Posisi Pemain (a) tingkat kesulitan *easy*. (b) tingkat kesulitan *medium*. (c) tingkat kesulitan *hard*. (d) tingkat kesulitan *insane*.

Posisi musuh ditunjukan pada variabel *End Node* dan posisi pemain ditunjukan pada variabel *Start Node P1* dan *Start Node P2* untuk pemain satu dan pemain dua.

5.2.4 Uji Coba Musuh Mencari Jalan Tercepat Menuju Pemain

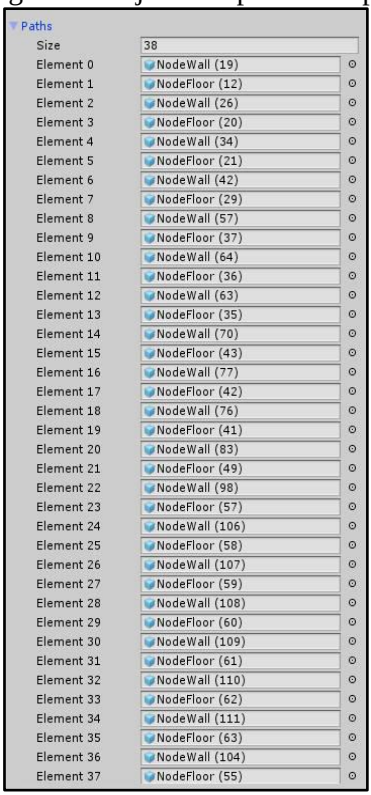
Pada sub bab ini dijelaskan secara detail mengenai skenario yang dilakukan dan hasil yang didapatkan dari pengujian fungsionalitas musuh dalam mencari jalan tercepat menuju pemain. Penjelasan disajikan dengan menampilkan kondisi awal, masukan, keluaran, hasil yang dicapai, dan kondisi akhir. Skenario yang telah diuji terdapat pada **Tabel 5.5**.

Tabel 5.5 Hasil Uji Coba Musuh Mencari Jalan Tercepat Menuju Pemain

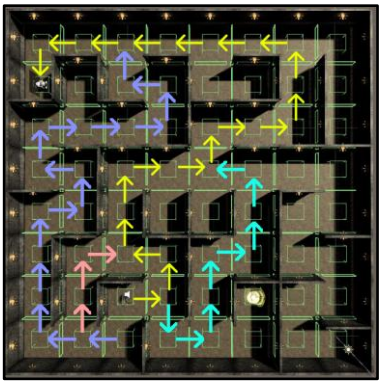
ID	UF-004
Nama	Uji coba musuh mencari jalan tercepat menuju pemain
Tujuan uji coba	Musuh menemukan jalan tercepat menuju pemain di dalam *maze*
Kondisi Awal	Pemain dan musuh berada di dalam *maze*
Skenario 1	***Musuh pada maze tingkat kesulitan easy mencari jalan tercepat menuju pemain***
Masukan	Posisi pemain dan posisi musuh
Keluaran yang diharapkan	Musuh menemukan pemain
Hasil Uji	Musuh pada *maze* tingkat kesulitan *easy* berhasil menemukan pemain
Skenario 2	***Musuh pada maze tingkat kesulitan medium mencari jalan tercepat menuju pemain***
Masukan	Posisi pemain dan posisi musuh
Keluaran yang diharapkan	Musuh menemukan pemain
Hasil Uji	Musuh pada *maze* tingkat kesulitan *medium* berhasil menemukan pemain
Skenario 3	***Musuh pada maze tingkat kesulitan hard mencari jalan tercepat menuju pemain***
Masukan	Posisi pemain dan posisi musuh
Keluaran yang diharapkan	Musuh menemukan pemain
Hasil Uji	Musuh pada *maze* tingkat kesulitan *hard* berhasil menemukan pemain
Skenario 4	***Musuh pada maze tingkat kesulitan insane mencari jalan tercepat menuju pemain***

Masukan	Posisi pemain dan posisi musuh
Keluaran yang diharapkan	Musuh menemukan pemain
Hasil Uji	Musuh pada *maze* tingkat kesulitan *insane* berhasil menemukan pemain

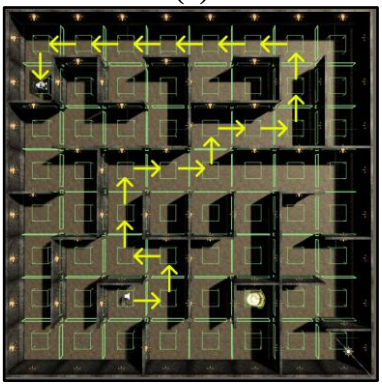
Hasil uji dari musuh menemukan pemain menggunakan algoritma Dijkstra dapat dilihat pada **Gambar 5.7**.



(a)



(b)



(c)

Gambar 5.7 Hasil Uji Coba Musuh Mencari Jalan Menuju Pemain (a) *Nodes* yang telah dipilih. (b) Visualisasi kemungkinan *paths*. (c) Visualisasi *path* dari *nodes* yang telah dipilih.

5.2.5 Uji Coba Musuh Menyerang Pemain

Pada sub bab ini dijelaskan secara detail mengenai skenario yang dilakukan dan hasil yang didapatkan dari pengujian fungsionalitas musuh dalam menyerang pemain. Penjelasan disajikan dengan menampilkan kondisi awal, masukan, keluaran, hasil yang dicapai, dan kondisi akhir. Skenario yang telah diuji terdapat pada **Tabel 5.6**.

Tabel 5.6 Hasil Uji Coba Musuh Menyerang Pemain

ID	UF-005
Nama	Uji coba musuh menyerang pemain
Tujuan uji coba	Musuh dapat menyerang pemain dalam *maze*
Kondisi Awal	Pemain dan musuh berada dalam *maze*
Skenario 1	***Musuh pada maze tingkat kesulitan easy menyerang pemain***
Masukan	Posisi musuh sangat dekat dengan posisi pemain
Keluaran yang diharapkan	Musuh melakukan serangan kepada pemain dan *health bar* pemain berkurang
Hasil Uji	Musuh pada *maze* tingkat kesulitan *easy* berhasil menyerang pemain
Skenario 2	***Musuh pada maze tingkat kesulitan medium menyerang pemain***
Masukan	Posisi musuh sangat dekat dengan posisi pemain
Keluaran yang diharapkan	Musuh melakukan serangan kepada pemain dan *health bar* pemain berkurang
Hasil Uji	Musuh pada *maze* tingkat kesulitan *medium* berhasil menyerang pemain
Skenario 3	***Musuh pada maze tingkat kesulitan hard menyerang pemain***

Masukan	Posisi musuh sangat dekat dengan posisi pemain
Keluaran	Musuh melakukan serangan kepada pemain dan *health bar* pemain berkurang
Hasil Uji	Musuh pada *maze* tingkat kesulitan *hard* berhasil menyerang pemain
Skenario 4	***Musuh pada maze tingkat kesulitan insane menyerang pemain***
Masukan	Posisi musuh sangat dekat dengan posisi pemain
Keluaran	Musuh melakukan serangan kepada pemain dan *health bar* pemain berkurang
Hasil Uji	Musuh pada *maze* tingkat kesulitan *insane* berhasil menyerang pemain

5.2.6 Uji Coba Efektivitas Algoritma Dijkstra

Pada sub bab ini akan dilakukan pembuktian bahwa algoritma Dijkstra efektif digunakan sebagai *artificial intelligence pathfinding* pada permainan *maze*. Uji coba dilakukan pada setiap tingkat kesulitan sebanyak tiga kali percobaan dan dihitung rata-rata waktunya. Hasil uji coba efektivitas algoritma Dijkstra pada tingkat kesulitan dapat dilihat pada tabel **Tabel 5.7**.

Tabel 5.7 Hasil Uji Coba Efektivitas Algoritma Dijkstra Tingkat Kesulitan Berdasarkan Waktu

Tingkat Kesulitan	Percobaan			Rata-rata (detik)
	1 (detik)	2 (detik)	3 (detik)	
Easy	277,74	178,26	234,79	230,26
Medium	174,51	183,61	135,48	164,53
Hard	151,11	124,72	130,77	135,53
Insane	121,12	121,27	121,03	121,14

5.2.7 Hasil Uji Coba

Pada sub bab ini diberikan hasil evaluasi dari pengujian yang dilakukan musuh pada *game* 'AVIAR'. Hasil evaluasi dapat dilihat pada **Tabel 5.8**.

Tabel 5.8 Hasil Uji Coba

ID	Deskripsi	Skenario	Perilaku Terlaksana
UF-001	Uji coba musuh muncul dalam *maze*	Skenario 1	Ya
UF-002	Uji coba musuh mengetahui kondisi *node* pada *maze*	Skenario 1	Ya
		Skenario 2	Ya
		Skenario 3	Ya
		Skenario 4	Ya
UF-003	Uji coba musuh mengetahui posisi pemain	Skenario 1	Ya
		Skenario 2	Ya
		Skenario 3	Ya
		Skenario 4	Ya
UF-004	Uji coba musuh mencari jalan tercepat menuju pemain	Skenario 1	Ya
		Skenario 2	Ya
		Skenario 3	Ya
		Skenario 4	Ya
UF-005	Uji coba musuh menyerang pemain	Skenario 1	Ya
		Skenario 2	Ya
		Skenario 3	Ya
		Skenario 4	Ya

5.3 Pengujian Pengguna

Selain melakukan pengujian fungsionalitas, dilakukan juga pengujian yang melibatkan pengguna secara langsung. Pengujian ini bernilai subjektif yang bertujuan untuk mengetahui tingkat keberhasilan *artificial intelligence* yang dibangun dari sudut pandang pengguna. Penilaian ini bisa didapat dengan meminta penilaian dan tanggapan dari pengguna terhadap aspek-aspek pada *artificial intelligence*.

5.3.1 Skenario Pengujian Pengguna

Dalam melakukan pengujian permainan, pengguna diminta mencoba memainkan permainan untuk mencoba semua fungsionalitas dan fitur yang ada. Pengujian permainan oleh pengguna dilakukan dengan memberikan informasi terlebih dahulu mengenai permainan, kegunaan, dan fitur-fitur yang dimiliki. Setelah informasi tersampaikan, pengguna kemudian diarahkan untuk langsung mencoba permainan dengan spesifikasi lingkungan yang sama dengan yang telah diuraikan pada **Tabel 5.1**.

Jumlah pengguna yang terlibat dalam pengujian permainan sebanyak 7 orang. Dalam melakukan pengujian, pengguna melakukan percobaan lebih dari satu kali penggunaan untuk masing-masing pengguna. Dalam memberikan penilaian dan tanggapan, pengguna diberikan kuesioner pengujian permainan. Kuesioner pengujian permainan ini memiliki beberapa aspek penilaian seputar desain antarmuka, *immersive*, *artificial intelligence*, *minigame*, *gameplay* dan tingkat kenyamanan permainan. Nilai yang diberikan memiliki rentang nilai 1 hingga 6 dengan rincian pada **Tabel 5.9**. Pada bagian akhir terdapat kritik dan saran untuk perbaikan fitur. Detail kuesioner pengguna dapat dilihat pada **Tabel 5.10**.

Tabel 5.9 Rentang Nilai

No.	Keterangan	Nilai
1	Sangat Tidak Setuju (STS)	1
2	Tidak Setuju (TS)	2
3	Kurang Setuju (KS)	3
4	Cukup Setuju (CS)	4
5	Setuju (S)	5
6	Sangat Setuju (SS)	6

Tabel 5.10 Kuesioner Pengguna

No.	Karakteristik Pemain
1	Apakah anda mengetahui *Virtual Reality*?
2	Apakah anda mengetahui permainan labirin (*maze game*)?
3	Apakah anda pernah bermain permainan labirin?

4	Apakah anda pernah bemain permainan labirin berbasis *Virtual Reality*?
	Jika pernah, Bagaimana mendapat anda?
5	Apakah anda mengetahui *Artificial Intelligence*?

No.	Parameter Antar Muka	STS	TS	KS	CS	S	SS
1.	Aplikasi memiliki tampilan, warna, dan desain yang menarik.						
2.	Aplikasi ini memiliki tata letak tombol, instruksi dan informasi yang mudah dipahami						
No.	Parameter Immersive	STS	TS	KS	CS	S	SS
3.	Anda merasakan sensasi nyata seperti di dalam *maze* sungguhan						
4.	Anda merasa tertantang untuk segera mencari jalan keluar *maze*						
5.	Anda merasakan suasana misterius yang ditawarkan oleh permainan						
No.	Parameter *Artificial Intelligence*	STS	TS	KS	CS	S	SS
6	Musuh dalam permainan ini dapat menemukan anda						
7	Semakin tinggi level, maka semakin cepat musuh menemukan anda						
8	Tingkat kesulitan musuh sesuai dengan						

	level permainan						
No.	Parameter *Minigame*	STS	TS	KS	CS	S	SS
9	*Minigame* Legot dapat melatih kemampuan logika anda						
10	*Minigame* Legot pada level *easy* memiliki tingkat kesulitan yang sesuai						
11	*Minigame* Legot pada level *medium* memiliki tingkat kesulitan yang sesuai						
12	*Minigame* Legot pada level *hard* memiliki tingkat kesulitan yang sesuai						
13	*Minigame* Picture Memory dapat melatih memori anda						
14	*Minigame* Picture Memory pada level *easy* memiliki tingkat kesulitan yang sesuai						
15	*Minigame* Picture Memory pada level *medium* memiliki tingkat kesulitan yang sesuai						
16	Minigame Picture Memory pada level *hard* memiliki tingkat kesulitan yang sesuai						
17	*Minigame* Tap Fast dapat melatih kecepatan refleks anda						

18	*Minigame* Tap Fast pada level *easy* memiliki tingkat kesulitan yang sesuai						
19	*Minigame* Tap Fast pada level *medium* memiliki tingkat kesulitan yang sesuai						
20	*Minigame* Tap Fast pada level *hard* memiliki tingkat kesulitan yang sesuai						
No.	Parameter *Gameplay*	STS	TS	KS	CS	S	SS
21	Rancangan *random maze* sudah sesuai dengan tingkat kesulitan levelnya						
22	Saya merasa *game* ini cocok dimainkan secara *multiplayer*						
23	Saya merasa dengan bantuan *power up*, *game* ini menjadi menarik						
No.	Parameter Kenyamanan	STS	TS	KS	CS	S	SS
24	Aplikasi dapat berjalan lancar tanpa *lag* dan *crash*						
25	Kontrol pergerakan *player* tidak membingungkan.						
26	Saya merasa nyaman selama menggunakan aplikasi ini						

5.3.2 Daftar Penguji Permainan

Pada sub bab ini berisi daftar pengguna yang bertindak sebagai penguji coba *game* yang dibangun. Dalam pengujian ini tidak terdapat kriteria atau keahlian khusus yang harus dimiliki pengguna karena *game* ini ditunjukkan kepada berbagai kalangan pengguna baik yang suka bermain permainan ataupun tidak. Daftar nama penguji aplikasi ini dapat dilihat pada **Tabel 5.11**.

Tabel 5.11 Daftar Penguji Permainan

No.	Nama	Pekerjaan
1	Ghaly Aditya	Mahasiswa
2	Riansya Pamusti	Mahasiswa
3	Yusuf Dimas H.	Mahasiswa
4	Rahmat Rijal	Mahasiswa
5	Sultan Bonar M.	Mahasiswa
6	Ananda Ricky	Mahasiswa
7	Hari Setiawan	Mahasiswa

5.3.3 Hasil Pengujian Pengguna

Uji coba yang dilakukan terhadap beberapa pengguna memiliki beberapa aspek yang dipisahkan berdasarkan antarmuka, *immersive, artificial intelligence, minigame, gameplay* dan tingkat kenyamanan permainan. Akan tetapi, hasil pengujian pengguna yang ditunjukan berupa hasil pengujian parameter *artificial intelligence* yang dapat dilihat pada **Tabel 5.12**.

Sistem penilaian didasarkan pada skala penghitungan 1 – 6 dimana skala satu menunjukkan nilai terendah dan skala enam menunjukkan skala tertinggi. Penilaian akhir kemudian dilakukan dengan menjumlahkan banyak penguji yang memilih suatu skala tertentu dan kemudian dibagi dengan jumlah penguji. Hasil uji coba dipaparkan secara lengkap dapat dilihat pada **Tabel 5.13**.

Tabel 5.12 Hasil Pengujian Pengguna

No.	Pernyataan	STS	TS	KS	CS	S	SS
Parameter *Artificial Intelligence*							
1	Musuh dalam permainan ini dapat menemukan anda	0	0	1	1	5	0
2	Semakin tinggi level, maka semakin cepat musuh menemukan anda	0	0	1	2	2	2
3	Tingkat kesulitan musuh sesuai dengan level permainan	0	1	1	2	3	0

Tabel 5.13 Hasil Akhir Pengujian Pengguna

No	**Pernyataan**	**Rata-Rata**	**Total**	**Total (%)**
Parameter *Artificial Intelligence*				
1	Musuh dalam permainan ini dapat menemukan anda	4,57	4,43	73,83
2	Semakin tinggi level, maka semakin cepat musuh menemukan anda	4,71		
3	Tingkat kesulitan musuh sesuai dengan level permainan	4		

5.3.4 Kritik dan Saran Pengguna

Dalam memberikan penilaian dan tanggapan, pengguna diberikan kuesioner pengujian perangkat lunak. Kuesioner pengujian perangkat lunak ini terdapat bagian kritik dan saran

untuk perbaikan fitur kedepannya. Kritik dan saran pengguna dapat dilihat pada **Tabel 5.14**.

Tabel 5.14 Kritik dan Saran Pengguna

No.	Nama	Kritik dan Saran
1	Ghaly Aditya	Memperlihatkan *maze* untuk beberapa detik sebelum permainan dimulai sehingga pemain bisa mengetahui / mengingat jalan yang dituju.
2	Riansya Pamusti	Rotasi pemain dibuat halus agar tidak memusingkan (*controller*).
3	Yusuf Dimas H.	Dibuat menjadi lebih nyaman untuk dimainkan agar tidak pusing saat bermain.
4	Rahmat Rijal	*Gameplay* AVIAR sudah bagus tetapi kalau bisa *asset* untuk lingkungan *game*nya diperbagus lagi agar membuat pemain tertarik dan merasa nyaman.
5	Sultan Bonar M.	Kecepatan saat memakai *boots* terlalu cepat, bikin pusing. Saran dipelankan (jangan ekstrem).
6	Ananda Ricky	*Game*nya buat pusing, berikan penawar pusing.
7	Hari Setiawan	Pergerakan pemain terlalu cepat sehingga meninggalkan musuh terlalu jauh. *Sound effect* kurang menarik.

5.4 Evaluasi Pengujian

Sub bab ini membahas mengenai evaluasi terhadap pengujian yang telah dilakukan yaitu pengujian fungsionalitas dan pengujian pengguna.

Berdasarkan hasil pengujian fungsionalitas yaitu memunculkan musuh, mengetahui kondisi *node*, mengetahui posisi pemain, mencari jalan tercepat, dan menyerang musuh yang dapat dilihat pada **Tabel 5.8** dapat disimpulkan bahwa musuh

telah berfungsi dengan baik dan algoritma *Dijkstra* efektif dan efisien dalam menemukan jalan tercepat dengan waktu pada tiap tingkat kesulitan dapat dilihat pada **Error! Reference source not found.**.

Berdasarkan hasil pengujian pengguna yang dapat dilihat pada **Tabel 5.13** bahwa *Artificial Intelligence* berhasil diimplementasikan pada permainan *maze* 'AVIAR' walaupun tingkat kesulitan musuh perlu lebih disesuaikan pada masing-masing *level maze*.

[Halaman ini sengaja dikosongkan]

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang dapat diambil dari hasil uji coba yang telah dilakukan sebagai jawaban dari rumusan masalah. Selain itu juga terdapat saran yang ditujukan untuk pengembangan penelitian lebih lanjut.

6.1 Kesimpulan

Kesimpulan yang diperoleh dari uji coba dan evaluasi adalah sebagai berikut:

1. Algoritma Dijkstra *pathfinding* dapat diimplementasikan pada *game* ‘AVIAR’.
2. Berdasarkan uji coba efektivitas yang dilakukan, Algoritma Dijkstra *pathfinding* dapat mencari jalan tercepat pada setiap tingkat kesulitan dengan rata-rata waktu yaitu *easy* 230,26 detik, *medium* 164,53 detik, *hard* 135,53 detik dan *insane* 121,14 detik.

6.2 Saran

Berikut saran untuk pengembangan dan perbaikan sistem di masa yang akan datang. Diantaranya dalah sebagai berikut:

1. Melakukan uji coba dengan menggunakan algoritma *pathfinding* lainnya.
2. Melakukan metode optimasi untuk *shortest path*.

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

[1] F. P. Brooks Jr., "What's Real About," *IEEE Computer Graphics And Applications,* 1999.

[2] Kickstarter, "Oculus Rift: Step Into the Game," 2018. [Online]. Available: https://www.kickstarter.com/projects/1523379957/oculus-rift-step-into-the-game. [Accessed 4 March 2018].

[3] A. A. Yunanto, D. Herumurti, and I. Kuswardayan, "Kecerdasan Buatan Pada Game Edukasi Untuk Pembelajaran Bahasa Inggris Berbasis Pendekatan Heuristik Similaritas," J. Sist. Dan Inform., 2017.

[4] M. Abdi, D. Herumurti, and I. Kuswardayan, "Analisis Perbandingan Kecerdasan Buatan Pada Computer Player dalam Mengambil Keputusan Pada Game Battle RPG," Ilm. Teknol. Inf., 2017.

[5] X. Cui and H. Shi, "An Overview of Pathfinding in Navigation Mesh," *IJCSNS International Journal of Computer Science and Network Security,* 2012.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest dan C. Stein, Introduction to Algorithms, United States: MIT Press, 1990.

[7] "Unity Technologies," Unity - Game Engine, 9 December 2017. [Online]. Available: http://unity3d.com/unity. [Accessed 23 December 2017].

[8] A. Hejlsberg, The A-Z of Programming Languages, 2008.

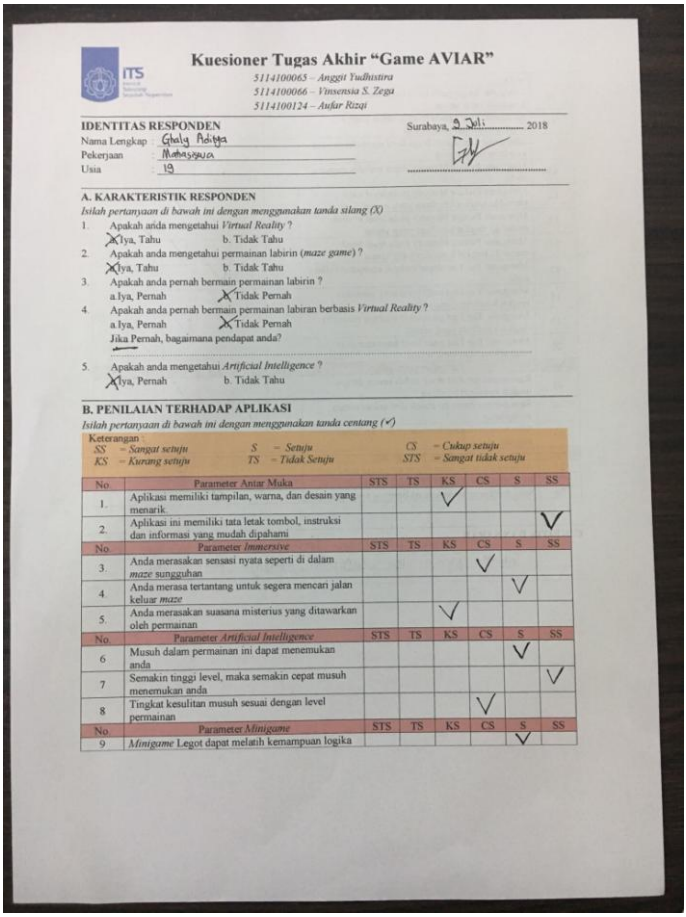
[*Halaman ini sengaja dikosongkan*]

LAMPIRAN

A. Hasil Kuesioner

1. Kuesioner dari Penguji Ke-1 (Ghaly Aditya)

a) Halaman Pertama



Kuesioner Tugas Akhir "Game AVIAR"

5114100065 – Anggit Yudhistira
5114100066 – Vinsensia S. Zega
5114100124 – Aufar Rizqi

IDENTITAS RESPONDEN

Nama Lengkap : Ghaly Aditya
Pekerjaan : Mahasiswa
Usia : 19

Surabaya, 9 Juli 2018

A. KARAKTERISTIK RESPONDEN

Isilah pertanyaan di bawah ini dengan menggunakan tanda silang (X)

1. Apakah anda mengetahui *Virtual Reality* ?
   X a. Iya, Tahu b. Tidak Tahu
2. Apakah anda mengetahui permainan labirin (*maze game*) ?
   X a. Iya, Tahu b. Tidak Tahu
3. Apakah anda pernah bermain permainan labirin ?
   a. Iya, Pernah X b. Tidak Pernah
4. Apakah anda pernah bermain permainan labiran berbasis *Virtual Reality* ?
   a. Iya, Pernah X b. Tidak Pernah
   Jika Pernah, bagaimana pendapat anda?
   -
5. Apakah anda mengetahui *Artificial Intelligence* ?
   X a. Iya, Pernah b. Tidak Tahu

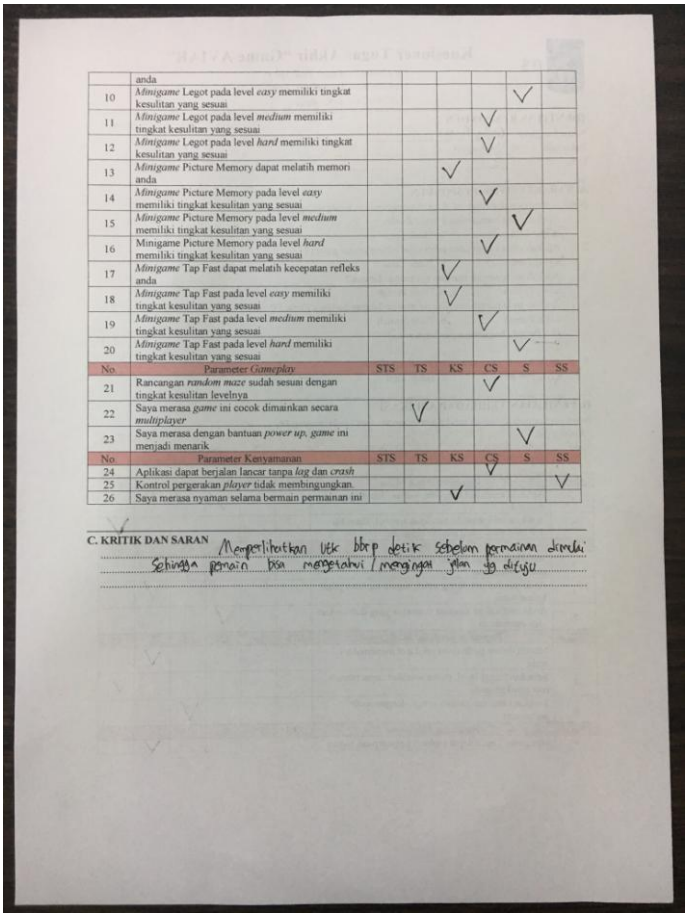
B. PENILAIAN TERHADAP APLIKASI

Isilah pertanyaan di bawah ini dengan menggunakan tanda centang (✓)

Keterangan :
SS = Sangat setuju *S = Setuju* *CS = Cukup setuju*
KS = Kurang setuju *TS = Tidak Setuju* *STS = Sangat tidak setuju*

No.	Parameter Antar Muka	STS	TS	KS	CS	S	SS
1.	Aplikasi memiliki tampilan, warna, dan desain yang menarik.			✓			
2.	Aplikasi ini memiliki tata letak tombol, instruksi dan informasi yang mudah dipahami						✓
No.	Parameter *Immersive*	STS	TS	KS	CS	S	SS
3.	Anda merasakan sensasi nyata seperti di dalam *maze* sungguhan				✓		
4.	Anda merasa tertantang untuk segera mencari jalan keluar *maze*					✓	
5.	Anda merasakan suasana misterius yang ditawarkan oleh permainan			✓			
No.	Parameter *Artificial Intelligence*	STS	TS	KS	CS	S	SS
6	Musuh dalam permainan ini dapat menemukan anda					✓	
7	Semakin tinggi level, maka semakin cepat musuh menemukan anda						✓
8	Tingkat kesulitan musuh sesuai dengan level permainan				✓		
No.	Parameter *Minigame*	STS	TS	KS	CS	S	SS
9	*Minigame* Legot dapat melatih kemampuan logika					✓	

b) Halaman Kedua

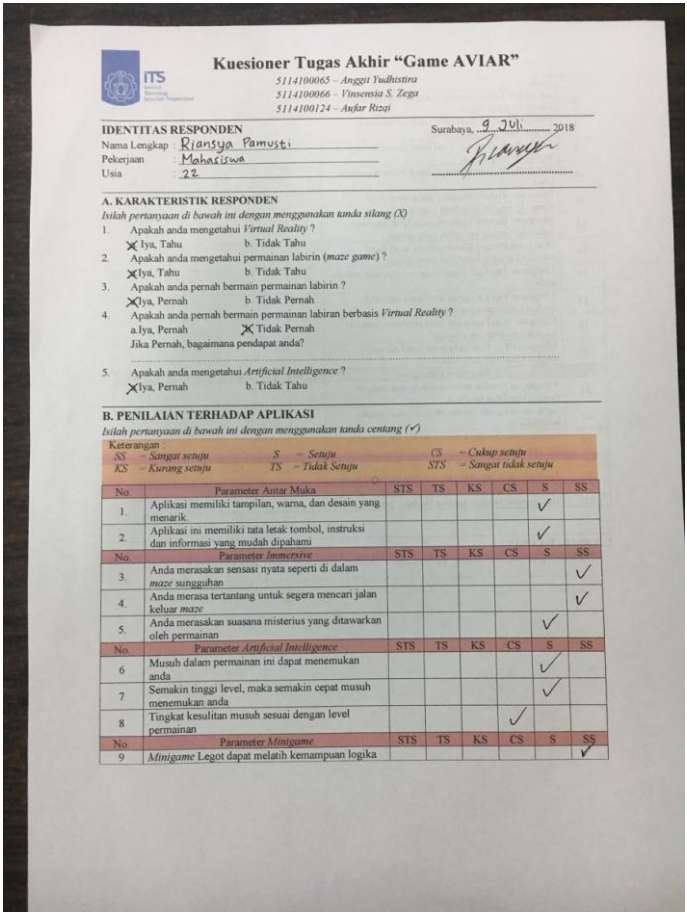


	anda						
10	*Minigame* Legot pada level *easy* memiliki tingkat kesulitan yang sesuai					V	
11	*Minigame* Legot pada level *medium* memiliki tingkat kesulitan yang sesuai				V		
12	*Minigame* Legot pada level *hard* memiliki tingkat kesulitan yang sesuai				V		
13	*Minigame* Picture Memory dapat melatih memori anda			V			
14	*Minigame* Picture Memory pada level *easy* memiliki tingkat kesulitan yang sesuai				V		
15	*Minigame* Picture Memory pada level *medium* memiliki tingkat kesulitan yang sesuai					V	
16	Minigame Picture Memory pada level *hard* memiliki tingkat kesulitan yang sesuai				V		
17	*Minigame* Tap Fast dapat melatih kecepatan refleks anda			V			
18	*Minigame* Tap Fast pada level *easy* memiliki tingkat kesulitan yang sesuai			V			
19	*Minigame* Tap Fast pada level *medium* memiliki tingkat kesulitan yang sesuai				V		
20	*Minigame* Tap Fast pada level *hard* memiliki tingkat kesulitan yang sesuai					V	
No	Parameter *Gameplay*	STS	TS	KS	CS	S	SS
21	Rancangan *random maze* sudah sesuai dengan tingkat kesulitan levelnya				V		
22	Saya merasa *game* ini cocok dimainkan secara *multiplayer*		V				
23	Saya merasa dengan bantuan *power up*, *game* ini menjadi menarik					V	
No	Parameter Kenyamanan	STS	TS	KS	CS	S	SS
24	Aplikasi dapat berjalan lancar tanpa *lag* dan *crash*				V		
25	Kontrol pergerakan *player* tidak membingungkan.						V
26	Saya merasa nyaman selama bermain permainan ini			V			

C. KRITIK DAN SARAN Memperlihatkan ttk bbrp detik sebelum permainan dimulai Sehingga pemain bisa mempelajari / mengingat jalan yg dilalui

2. Kuesioner dari Penguji Ke-2 (Riansya Pamusti)

a) Halaman Pertama



Kuesioner Tugas Akhir "Game AVIAR"

5114100065 – Anggit Yudhistira
5114100066 – Vinsensia S. Zega
5114100124 – Aufar Rizqi

IDENTITAS RESPONDEN

Nama Lengkap : Riansya Pamusti
Pekerjaan : Mahasiswa
Usia : 22

Surabaya, 9 Juli 2018

A. KARAKTERISTIK RESPONDEN

Isilah pertanyaan di bawah ini dengan menggunakan tanda silang (X)

1. Apakah anda mengetahui *Virtual Reality* ?
   X a. Iya, Tahu b. Tidak Tahu
2. Apakah anda mengetahui permainan labirin (*maze game*) ?
   X a. Iya, Tahu b. Tidak Tahu
3. Apakah anda pernah bermain permainan labirin ?
   X a. Iya, Pernah b. Tidak Pernah
4. Apakah anda pernah bermain permainan labiran berbasis *Virtual Reality* ?
   a. Iya, Pernah X b. Tidak Pernah
   Jika Pernah, bagaimana pendapat anda?
   .....
5. Apakah anda mengetahui *Artificial Intelligence* ?
   X a. Iya, Pernah b. Tidak Tahu

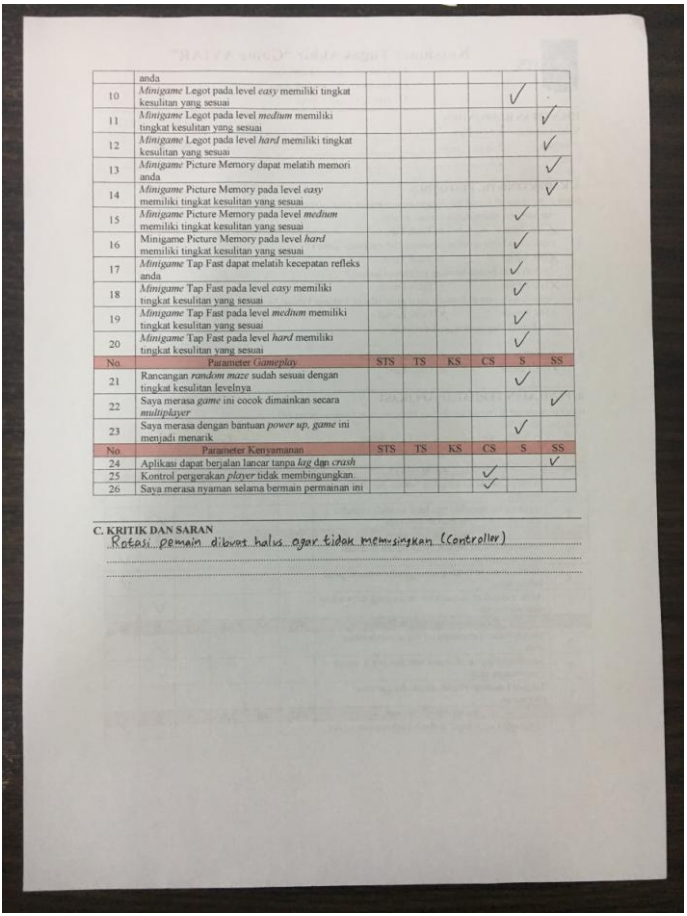
B. PENILAIAN TERHADAP APLIKASI

Isilah pertanyaan di bawah ini dengan menggunakan tanda centang (✓)

Keterangan :
SS = Sangat setuju *S = Setuju* *CS = Cukup setuju*
KS = Kurang setuju *TS = Tidak Setuju* *STS = Sangat tidak setuju*

No.	Parameter Antar Muka	STS	TS	KS	CS	S	SS
1.	Aplikasi memiliki tampilan, warna, dan desain yang menarik					✓	
2.	Aplikasi ini memiliki tata letak tombol, instruksi dan informasi yang mudah dipahami					✓	
No.	Parameter *Immersive*	STS	TS	KS	CS	S	SS
3.	Anda merasakan sensasi nyata seperti di dalam *maze* sungguhan						✓
4.	Anda merasa tertantang untuk segera mencari jalan keluar *maze*						✓
5.	Anda merasakan suasana misterius yang ditawarkan oleh permainan					✓	
No.	Parameter *Artificial Intelligence*	STS	TS	KS	CS	S	SS
6	Musuh dalam permainan ini dapat menemukan anda					✓	
7	Semakin tinggi level, maka semakin cepat musuh menemukan anda					✓	
8	Tingkat kesulitan musuh sesuai dengan level permainan				✓		
No.	Parameter *Minigame*	STS	TS	KS	CS	S	SS
9	*Minigame* Legot dapat melatih kemampuan logika						✓

b) Halaman Kedua



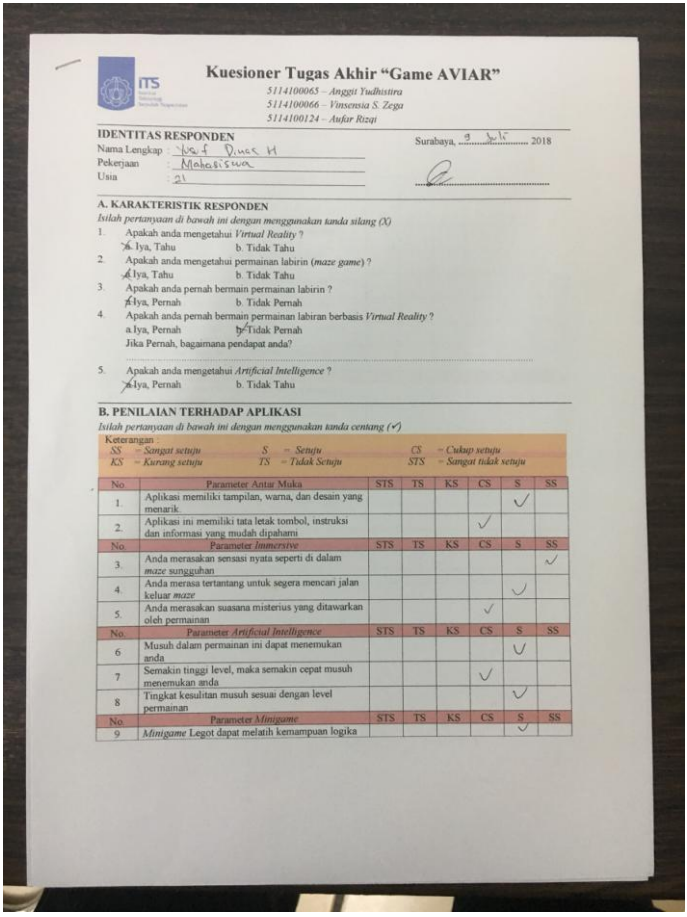
	anda						
10	*Minigame* Legot pada level *easy* memiliki tingkat kesulitan yang sesuai					✓	
11	*Minigame* Legot pada level *medium* memiliki tingkat kesulitan yang sesuai						✓
12	*Minigame* Legot pada level *hard* memiliki tingkat kesulitan yang sesuai						✓
13	*Minigame* Picture Memory dapat melatih memori anda						✓
14	*Minigame* Picture Memory pada level *eazy* memiliki tingkat kesulitan yang sesuai						✓
15	*Minigame* Picture Memory pada level *medium* memiliki tingkat kesulitan yang sesuai					✓	
16	Minigame Picture Memory pada level *hard* memiliki tingkat kesulitan yang sesuai					✓	
17	*Minigame* Tap Fast dapat melatih kecepatan refleks anda					✓	
18	*Minigame* Tap Fast pada level *easy* memiliki tingkat kesulitan yang sesuai					✓	
19	*Minigame* Tap Fast pada level *medium* memiliki tingkat kesulitan yang sesuai					✓	
20	*Minigame* Tap Fast pada level *hard* memiliki tingkat kesulitan yang sesuai					✓	
No	Parameter *Gameplay*	STS	TS	KS	CS	S	SS
21	Rancangan *random maze* sudah sesuai dengan tingkat kesulitan levelnya					✓	
22	Saya merasa *game* ini cocok dimainkan secara *multiplayer*						✓
23	Saya merasa dengan bantuan *power up*, *game* ini menjadi menarik					✓	
No	Parameter Kenyamanan	STS	TS	KS	CS	S	SS
24	Aplikasi dapat berjalan lancar tanpa *lag* dan *crash*						✓
25	Kontrol pergerakan *player* tidak membingungkan.				✓		
26	Saya merasa nyaman selama bermain permainan ini				✓		

C. KRITIK DAN SARAN

Rotasi pemain dibuat halus agar tidak membingkan (Controller)

3. Kuesioner dari Penguji Ketiga (Yusuf Dimas H.)

a) Halaman Pertama



Kuesioner Tugas Akhir "Game AVIAR"

5114100065 – Anggit Yudhistira
5114100066 – Vinsensia S. Zega
5114100124 – Aufar Rizqi

IDENTITAS RESPONDEN

Nama Lengkap : Yusf Dimas H
Pekerjaan : Mahasiswa
Usia : 21

Surabaya, 9 Juli 2018

A. KARAKTERISTIK RESPONDEN

Isilah pertanyaan di bawah ini dengan menggunakan tanda silang (X)

1. Apakah anda mengetahui *Virtual Reality* ?
   a. Iya, Tahu b. Tidak Tahu
2. Apakah anda mengetahui permainan labirin (*maze game*) ?
   a. Iya, Tahu b. Tidak Tahu
3. Apakah anda pernah bermain permainan labirin ?
   a. Iya, Pernah b. Tidak Pernah
4. Apakah anda pernah bermain permainan labiran berbasis *Virtual Reality* ?
   a. Iya, Pernah b. Tidak Pernah
   Jika Pernah, bagaimana pendapat anda?
   .....
5. Apakah anda mengetahui *Artificial Intelligence* ?
   a. Iya, Pernah b. Tidak Tahu

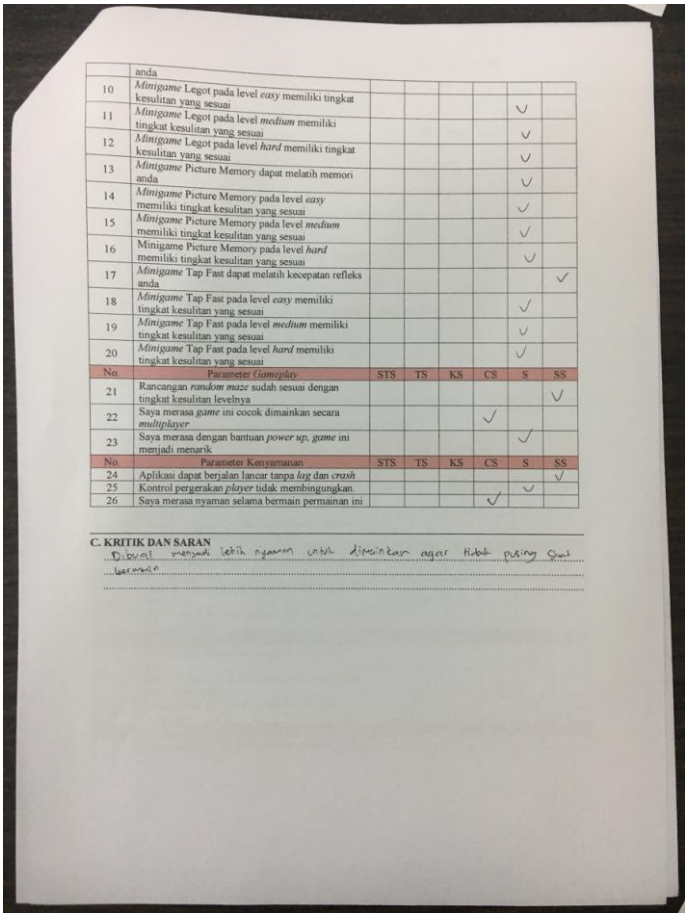
B. PENILAIAN TERHADAP APLIKASI

Isilah pertanyaan di bawah ini dengan menggunakan tanda centang (✓)

Keterangan :
SS = Sangat setuju *S = Setuju* *CS = Cukup setuju*
KS = Kurang setuju *TS = Tidak Setuju* *STS = Sangat tidak setuju*

No.	Parameter Antar Muka	STS	TS	KS	CS	S	SS
1.	Aplikasi memiliki tampilan, warna, dan desain yang menarik					✓	
2.	Aplikasi ini memiliki tata letak tombol, instruksi dan informasi yang mudah dipahami				✓		
No.	Parameter *Immersive*	STS	TS	KS	CS	S	SS
3.	Anda merasakan sensasi nyata seperti di dalam *maze* sungguhan						✓
4.	Anda merasa tertantang untuk segera mencari jalan keluar *maze*					✓	
5.	Anda merasakan suasana misterius yang ditawarkan oleh permainan				✓		
No.	Parameter *Artificial Intelligence*	STS	TS	KS	CS	S	SS
6	Musuh dalam permainan ini dapat menemukan anda					✓	
7	Semakin tinggi level, maka semakin cepat musuh menemukan anda				✓		
8	Tingkat kesulitan musuh sesuai dengan level permainan					✓	
No.	Parameter *Minigame*	STS	TS	KS	CS	S	SS
9	*Minigame* Legot dapat melatih kemampuan logika					✓	

b) Halaman Kedua



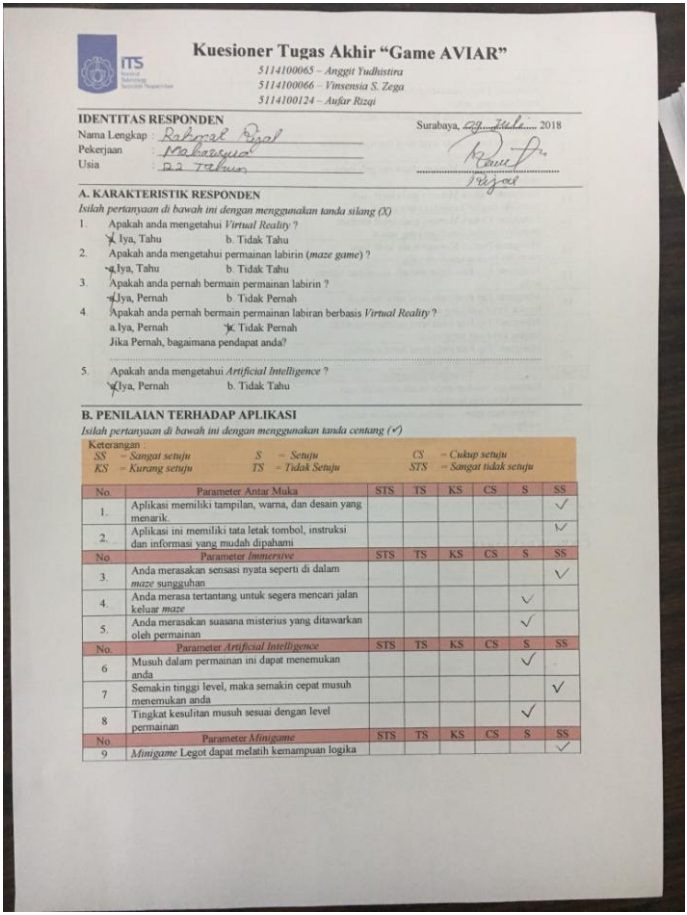
	anda						
10	*Minigame* Legot pada level *easy* memiliki tingkat kesulitan yang sesuai					✓	
11	*Minigame* Legot pada level *medium* memiliki tingkat kesulitan yang sesuai					✓	
12	*Minigame* Legot pada level *hard* memiliki tingkat kesulitan yang sesuai					✓	
13	*Minigame* Picture Memory dapat melatih memori anda					✓	
14	*Minigame* Picture Memory pada level *easy* memiliki tingkat kesulitan yang sesuai					✓	
15	*Minigame* Picture Memory pada level *medium* memiliki tingkat kesulitan yang sesuai					✓	
16	Minigame Picture Memory pada level *hard* memiliki tingkat kesulitan yang sesuai					✓	
17	*Minigame* Tap Fast dapat melatih kecepatan refleks anda						✓
18	*Minigame* Tap Fast pada level *easy* memiliki tingkat kesulitan yang sesuai					✓	
19	*Minigame* Tap Fast pada level *medium* memiliki tingkat kesulitan yang sesuai					✓	
20	*Minigame* Tap Fast pada level *hard* memiliki tingkat kesulitan yang sesuai					✓	
No	Parameter *Gameplay*	STS	TS	KS	CS	S	SS
21	Rancangan *random maze* sudah sesuai dengan tingkat kesulitan levelnya						✓
22	Saya merasa *game* ini cocok dimainkan secara *multiplayer*				✓		
23	Saya merasa dengan bantuan *power up*, *game* ini menjadi menarik					✓	
No	Parameter Kenyamanan	STS	TS	KS	CS	S	SS
24	Aplikasi dapat berjalan lancar tanpa *lag* dan *crash*						✓
25	Kontrol pergerakan *player* tidak membingungkan					✓	
26	Saya merasa nyaman selama bermain permainan ini				✓		

C. KRITIK DAN SARAN

Dibuat menjadi lebih nyaman untuk dimainkan agar tidak pusing saat bermain.

4. Kuesioner dari Penguji Ke-4 (Rahmat Rijal)

a) Halaman Pertama



Kuesioner Tugas Akhir "Game AVIAR"

5114100065 – Anggit Yudhistira
5114100066 – Vinsensia S. Zega
5114100124 – Aufar Rizqi

IDENTITAS RESPONDEN

Nama Lengkap : Rahmat Rijal
Pekerjaan : Mahasiswa
Usia : 22 Tahun

Surabaya, 29 Juli 2018

Rahmat Rijal

A. KARAKTERISTIK RESPONDEN

Isilah pertanyaan di bawah ini dengan menggunakan tanda silang (X)

1. Apakah anda mengetahui *Virtual Reality* ?
   X a. Iya, Tahu b. Tidak Tahu
2. Apakah anda mengetahui permainan labirin (*maze game*) ?
   X a. Iya, Tahu b. Tidak Tahu
3. Apakah anda pernah bermain permainan labirin ?
   X a. Iya, Pernah b. Tidak Pernah
4. Apakah anda pernah bermain permainan labiran berbasis *Virtual Reality* ?
   a. Iya, Pernah X b. Tidak Pernah
   Jika Pernah, bagaimana pendapat anda?
   .....
5. Apakah anda mengetahui *Artificial Intelligence* ?
   X a. Iya, Pernah b. Tidak Tahu

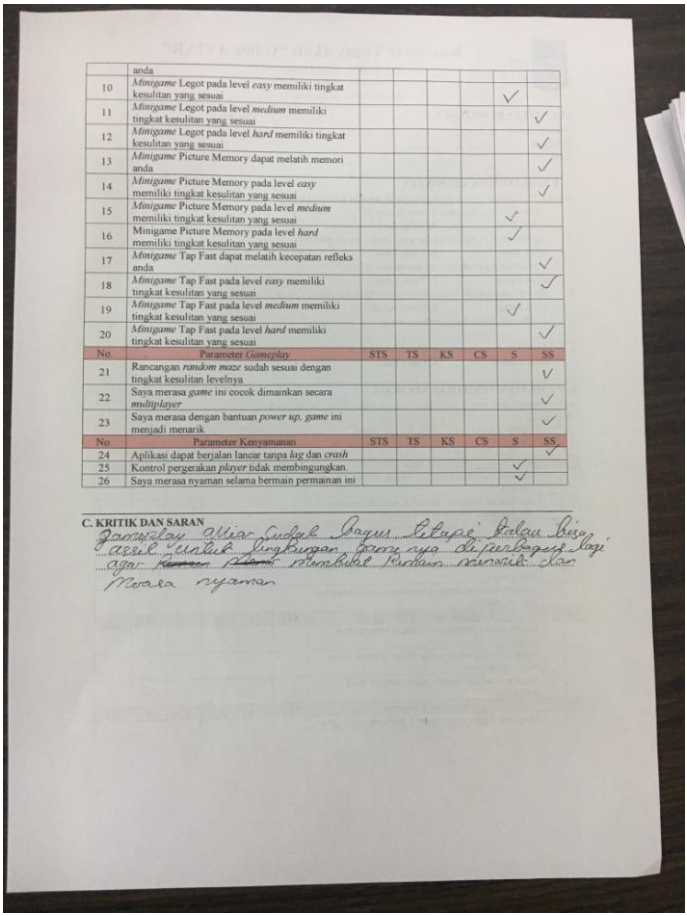
B. PENILAIAN TERHADAP APLIKASI

Isilah pertanyaan di bawah ini dengan menggunakan tanda centang (✓)

Keterangan :
SS = Sangat setuju *S = Setuju* *CS = Cukup setuju*
KS = Kurang setuju *TS = Tidak Setuju* *STS = Sangat tidak setuju*

No.	Parameter Antar Muka	STS	TS	KS	CS	S	SS
1.	Aplikasi memiliki tampilan, warna, dan desain yang menarik.						✓
2.	Aplikasi ini memiliki tata letak tombol, instruksi dan informasi yang mudah dipahami						✓
No.	Parameter *Immersive*	STS	TS	KS	CS	S	SS
3.	Anda merasakan sensasi nyata seperti di dalam *maze* sungguhan						✓
4.	Anda merasa tertantang untuk segera mencari jalan keluar *maze*					✓	
5.	Anda merasakan suasana misterius yang ditawarkan oleh permainan					✓	
No.	Parameter *Artificial Intelligence*	STS	TS	KS	CS	S	SS
6	Musuh dalam permainan ini dapat menemukan anda					✓	
7	Semakin tinggi level, maka semakin cepat musuh menemukan anda						✓
8	Tingkat kesulitan musuh sesuai dengan level permainan					✓	
No.	Parameter *Minigame*	STS	TS	KS	CS	S	SS
9	*Minigame* Legot dapat melatih kemampuan logika						✓

b) Halaman Kedua



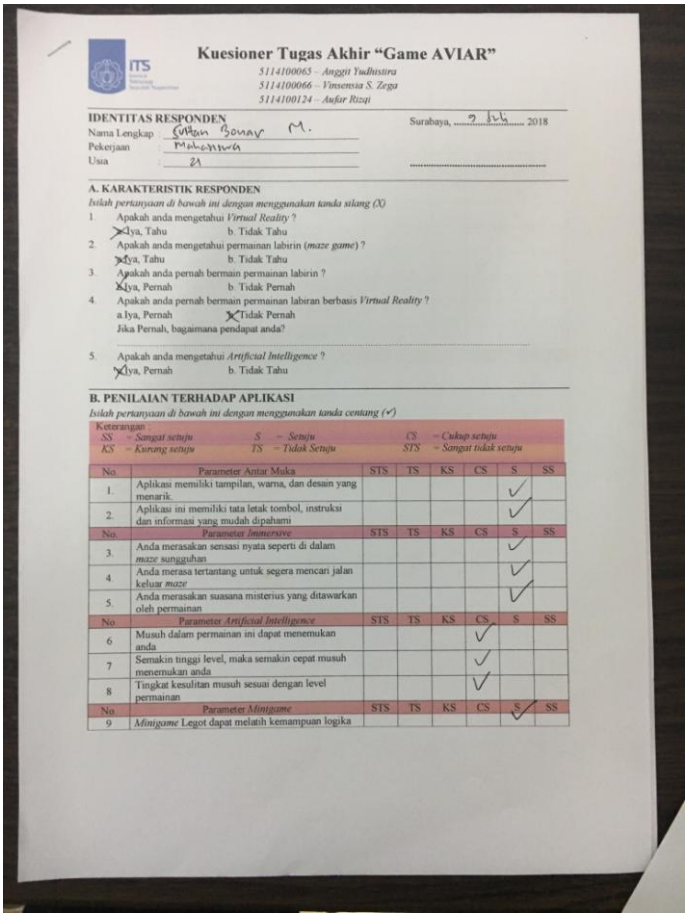
	anda						
10	*Minigame* Legot pada level *easy* memiliki tingkat kesulitan yang sesuai					✓	
11	*Minigame* Legot pada level *medium* memiliki tingkat kesulitan yang sesuai						✓
12	*Minigame* Legot pada level *hard* memiliki tingkat kesulitan yang sesuai						✓
13	*Minigame* Picture Memory dapat melatih memori anda						✓
14	*Minigame* Picture Memory pada level *easy* memiliki tingkat kesulitan yang sesuai						✓
15	*Minigame* Picture Memory pada level *medium* memiliki tingkat kesulitan yang sesuai					✓	
16	Minigame Picture Memory pada level *hard* memiliki tingkat kesulitan yang sesuai					✓	
17	*Minigame* Tap Fast dapat melatih kecepatan refleks anda						✓
18	*Minigame* Tap Fast pada level *easy* memiliki tingkat kesulitan yang sesuai						✓
19	*Minigame* Tap Fast pada level *medium* memiliki tingkat kesulitan yang sesuai					✓	
20	*Minigame* Tap Fast pada level *hard* memiliki tingkat kesulitan yang sesuai						✓
No	Parameter *Gameplay*	STS	TS	KS	CS	S	SS
21	Rancangan *random maze* sudah sesuai dengan tingkat kesulitan levelnya						✓
22	Saya merasa *game* ini cocok dimainkan secara *multiplayer*						✓
23	Saya merasa dengan bantuan *power up*, *game* ini menjadi menarik						✓
No	Parameter Kenyamanan	STS	TS	KS	CS	S	SS
24	Aplikasi dapat berjalan lancar tanpa *lag* dan *crash*						✓
25	Kontrol pergerakan *player* tidak membingungkan					✓	
26	Saya merasa nyaman selama bermain permainan ini					✓	

C. KRITIK DAN SARAN

Game nya sudah bagus, tetapi dalam beberapa level untuk lingkungan game nya di perbagus lagi agar ~~kamera~~ ~~dalam~~ membuat pemain merasa lebih nyaman

5. Kuesioner dari Penguji Ke-5 (Sultan Bonar M.)

a) Halaman Pertama



Kuesioner Tugas Akhir "Game AVIAR"

5114100065 – Anggit Yudhistira
5114100066 – Vinsensia S. Zega
5114100124 – Asfar Rizqi

IDENTITAS RESPONDEN

Nama Lengkap : Sultan Bonar M.
Pekerjaan : Mahasiswa
Usia : 21

Surabaya, 9 Juli 2018

A. KARAKTERISTIK RESPONDEN

Isikah pertanyaan di bawah ini dengan menggunakan tanda silang (X)

1. Apakah anda mengetahui *Virtual Reality* ?
   X a. Iya, Tahu b. Tidak Tahu
2. Apakah anda mengetahui permainan labirin (*maze game*) ?
   X a. Iya, Tahu b. Tidak Tahu
3. Apakah anda pernah bermain permainan labirin ?
   X a. Iya, Pernah b. Tidak Pernah
4. Apakah anda pernah bermain permainan labiran berbasis *Virtual Reality* ?
   a Iya, Pernah X Tidak Pernah
   Jika Pernah, bagaimana pendapat anda?
   .....
5. Apakah anda mengetahui *Artificial Intelligence* ?
   X a. Iya, Pernah b. Tidak Tahu

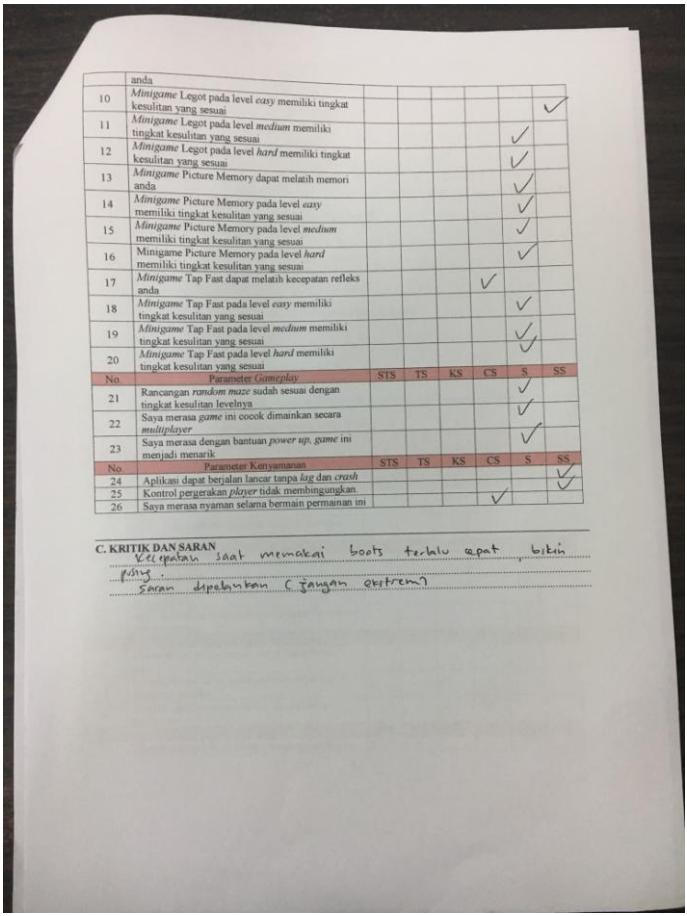
B. PENILAIAN TERHADAP APLIKASI

Isikah pertanyaan di bawah ini dengan menggunakan tanda centang (✓)

Keterangan :
SS = Sangat setuju S = Setuju CS = Cukup setuju
KS = Kurang setuju TS = Tidak Setuju STS = Sangat tidak setuju

No.	Parameter Antar Muka	STS	TS	KS	CS	S	SS
1.	Aplikasi memiliki tampilan, warna, dan desain yang menarik					✓	
2.	Aplikasi ini memiliki tata letak tombol, instruksi dan informasi yang mudah dipahami					✓	
No.	Parameter *Immersive*	STS	TS	KS	CS	S	SS
3.	Anda merasakan sensasi nyata seperti di dalam *maze* sungguhan					✓	
4.	Anda merasa tertantang untuk segera mencari jalan keluar *maze*					✓	
5.	Anda merasakan suasana misterius yang ditawarkan oleh permainan					✓	
No	Parameter *Artificial Intelligence*	STS	TS	KS	CS	S	SS
6	Musuh dalam permainan ini dapat menemukan anda				✓		
7	Semakin tinggi level, maka semakin cepat musuh menemukan anda				✓		
8	Tingkat kesulitan musuh sesuai dengan level permainan				✓		
No	Parameter *Minigame*	STS	TS	KS	CS	S	SS
9	*Minigame* Legot dapat melatih kemampuan logika					✓	

b) Halaman Kedua



	anda						
10	*Minigame* Legot pada level *easy* memiliki tingkat kesulitan yang sesuai						✓
11	*Minigame* Legot pada level *medium* memiliki tingkat kesulitan yang sesuai					✓	
12	*Minigame* Legot pada level *hard* memiliki tingkat kesulitan yang sesuai					✓	
13	*Minigame* Picture Memory dapat melatih memori anda					✓	
14	*Minigame* Picture Memory pada level *easy* memiliki tingkat kesulitan yang sesuai					✓	
15	*Minigame* Picture Memory pada level *medium* memiliki tingkat kesulitan yang sesuai					✓	
16	Minigame Picture Memory pada level *hard* memiliki tingkat kesulitan yang sesuai					✓	
17	*Minigame* Tap Fast dapat melatih kecepatan refleks anda				✓		
18	*Minigame* Tap Fast pada level *easy* memiliki tingkat kesulitan yang sesuai					✓	
19	*Minigame* Tap Fast pada level *medium* memiliki tingkat kesulitan yang sesuai					✓	
20	*Minigame* Tap Fast pada level *hard* memiliki tingkat kesulitan yang sesuai					✓	
No.	Parameter *Gameplay*	STS	TS	KS	CS	S	SS
21	Rancangan *random maze* sudah sesuai dengan tingkat kesulitan levelnya					✓	
22	Saya merasa *game* ini cocok dimainkan secara *multiplayer*					✓	
23	Saya merasa dengan bantuan *power up*, *game* ini menjadi menarik					✓	
No.	Parameter Kenyamanan	STS	TS	KS	CS	S	SS
24	Aplikasi dapat berjalan lancar tanpa *lag* dan *crash*						✓
25	Kontrol pergerakan *player* tidak membingungkan.						✓
26	Saya merasa nyaman selama bermain permainan ini				✓		

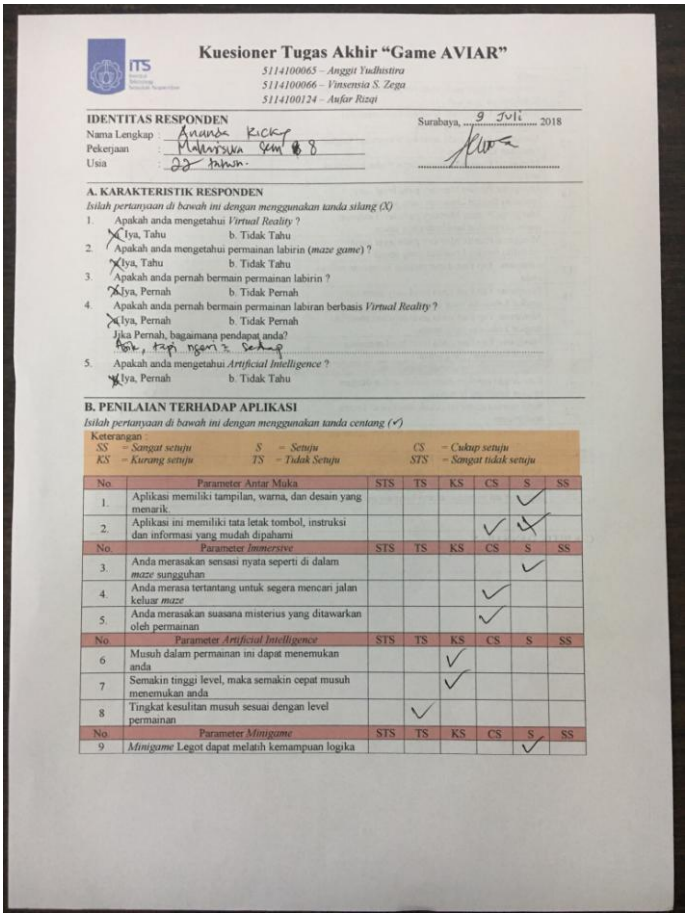
C. KRITIK DAN SARAN

Kelemahan saat memakai boots terlalu cepat bikin pusing.

Saran dipelankan (jangan ekstrem)

6. Kuesioner dari Penguji Ke-6 (Ananda Ricky)

a) Halaman Pertama



Kuesioner Tugas Akhir "Game AVIAR"

5114100065 – Anggit Yudhistira
5114100066 – Vinsensia S. Zega
5114100124 – Aufar Rizqi

IDENTITAS RESPONDEN

Nama Lengkap : Ananda Ricky
Pekerjaan : Mahasiswa Sem 8
Usia : 22 tahun.

Surabaya, 9 Juli 2018

A. KARAKTERISTIK RESPONDEN

Isilah pertanyaan di bawah ini dengan menggunakan tanda silang (X)

1. Apakah anda mengetahui *Virtual Reality* ?
   X a. Iya, Tahu b. Tidak Tahu
2. Apakah anda mengetahui permainan labirin (*maze game*) ?
   X a. Iya, Tahu b. Tidak Tahu
3. Apakah anda pernah bermain permainan labirin ?
   X a. Iya, Pernah b. Tidak Pernah
4. Apakah anda pernah bermain permainan labiran berbasis *Virtual Reality* ?
   X a. Iya, Pernah b. Tidak Pernah
   Jika Pernah, bagaimana pendapat anda?
   Fun, tapi ngeri sedikit
5. Apakah anda mengetahui *Artificial Intelligence* ?
   X a. Iya, Pernah b. Tidak Tahu

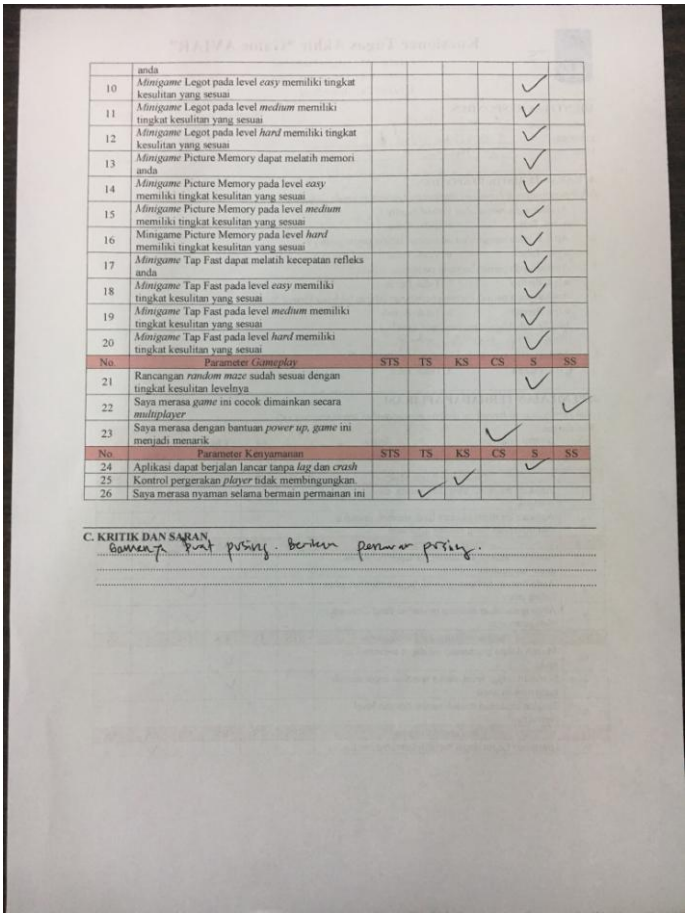
B. PENILAIAN TERHADAP APLIKASI

Isilah pertanyaan di bawah ini dengan menggunakan tanda centang (✓)

Keterangan
SS = *Sangat setuju* *S* = *Setuju* *CS* = *Cukup setuju*
KS = *Kurang setuju* *TS* = *Tidak Setuju* *STS* = *Sangat tidak setuju*

No	Parameter Antar Muka	STS	TS	KS	CS	S	SS
1.	Aplikasi memiliki tampilan, warna, dan desain yang menarik.					✓	
2.	Aplikasi ini memiliki tata letak tombol, instruksi dan informasi yang mudah dipahami				✓	✓	
No.	Parameter *Immersive*	STS	TS	KS	CS	S	SS
3.	Anda merasakan sensasi nyata seperti di dalam *maze* sungguhan					✓	
4.	Anda merasa tertantang untuk segera mencari jalan keluar *maze*				✓		
5.	Anda merasakan suasana misterius yang ditawarkan oleh permainan				✓		
No.	Parameter *Artificial Intelligence*	STS	TS	KS	CS	S	SS
6	Musuh dalam permainan ini dapat menemukan anda			✓			
7	Semakin tinggi level, maka semakin cepat musuh menemukan anda			✓			
8	Tingkat kesulitan musuh sesuai dengan level permainan		✓				
No.	Parameter *Minigame*	STS	TS	KS	CS	S	SS
9	*Minigame* Legot dapat melatih kemampuan logika					✓	

b) Halaman Kedua



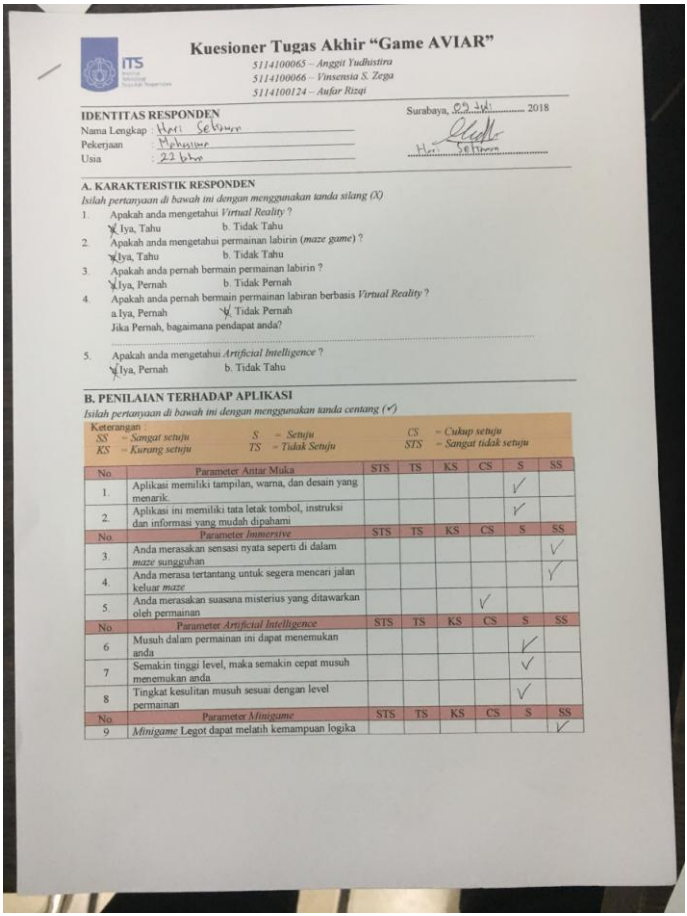
	anda						
10	*Minigame* Legot pada level *easy* memiliki tingkat kesulitan yang sesuai					✓	
11	*Minigame* Legot pada level *medium* memiliki tingkat kesulitan yang sesuai					✓	
12	*Minigame* Legot pada level *hard* memiliki tingkat kesulitan yang sesuai					✓	
13	*Minigame* Picture Memory dapat melatih memori anda					✓	
14	*Minigame* Picture Memory pada level *easy* memiliki tingkat kesulitan yang sesuai					✓	
15	*Minigame* Picture Memory pada level *medium* memiliki tingkat kesulitan yang sesuai					✓	
16	Minigame Picture Memory pada level *hard* memiliki tingkat kesulitan yang sesuai					✓	
17	*Minigame* Tap Fast dapat melatih kecepatan refleks anda					✓	
18	*Minigame* Tap Fast pada level *easy* memiliki tingkat kesulitan yang sesuai					✓	
19	*Minigame* Tap Fast pada level *medium* memiliki tingkat kesulitan yang sesuai					✓	
20	*Minigame* Tap Fast pada level *hard* memiliki tingkat kesulitan yang sesuai					✓	
No.	Parameter *Gameplay*	STS	TS	KS	CS	S	SS
21	Rancangan *random maze* sudah sesuai dengan tingkat kesulitan levelnya					✓	
22	Saya merasa *game* ini cocok dimainkan secara *multiplayer*						✓
23	Saya merasa dengan bantuan *power up*, *game* ini menjadi menarik				✓		
No	Parameter Kenyamanan	STS	TS	KS	CS	S	SS
24	Aplikasi dapat berjalan lancar tanpa *lag* dan *crash*					✓	
25	Kontrol pergerakan *player* tidak membingungkan.			✓			
26	Saya merasa nyaman selama bermain permainan ini		✓				

C. KRITIK DAN SARAN

Gamenya bikin pusing. Bentuk permen pusing.

7. Kuesioner dari Penguji Ke-7 (Hari Setiawan)

a) Halaman Pertama



Kuesioner Tugas Akhir "Game AVIAR"

5114100065 – Anggit Yudhistira
5114100066 – Vinsensia S. Zega
5114100124 – Aufar Rizqi

IDENTITAS RESPONDEN

Nama Lengkap : Hari Setiawan
Pekerjaan : Mahasiswa
Usia : 22 tahun

Surabaya, 03 Juli 2018

Hari Setiawan

A. KARAKTERISTIK RESPONDEN

Isilah pertanyaan di bawah ini dengan menggunakan tanda silang (X)

1. Apakah anda mengetahui *Virtual Reality* ?
   a. Iya, Tahu b. Tidak Tahu
2. Apakah anda mengetahui permainan labirin (*maze game*) ?
   a. Iya, Tahu b. Tidak Tahu
3. Apakah anda pernah bermain permainan labirin ?
   a. Iya, Pernah b. Tidak Pernah
4. Apakah anda pernah bermain permainan labiran berbasis *Virtual Reality* ?
   a. Iya, Pernah b. Tidak Pernah
   Jika Pernah, bagaimana pendapat anda?
   .....
5. Apakah anda mengetahui *Artificial Intelligence* ?
   a. Iya, Pernah b. Tidak Tahu

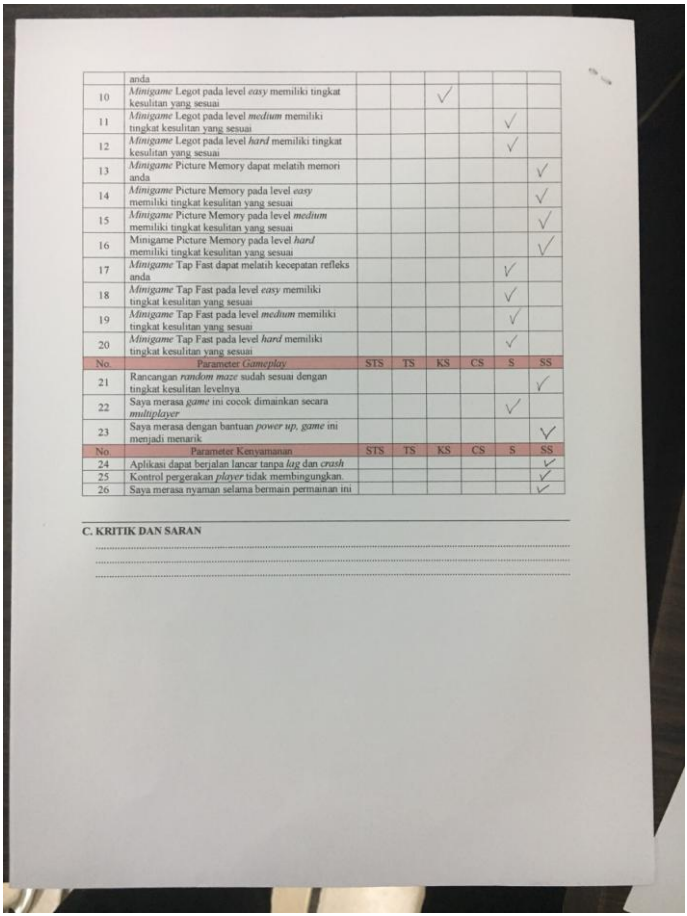
B. PENILAIAN TERHADAP APLIKASI

Isilah pertanyaan di bawah ini dengan menggunakan tanda centang (✓)

Keterangan :
SS = Sangat setuju *S = Setuju* *CS = Cukup setuju*
KS = Kurang setuju *TS = Tidak Setuju* *STS = Sangat tidak setuju*

No.	Parameter Antar Muka	STS	TS	KS	CS	S	SS
1.	Aplikasi memiliki tampilan, warna, dan desain yang menarik.					✓	
2.	Aplikasi ini memiliki tata letak tombol, instruksi dan informasi yang mudah dipahami					✓	
No.	Parameter *Immersive*	STS	TS	KS	CS	S	SS
3.	Anda merasakan sensasi nyata seperti di dalam *maze* sungguhan						✓
4.	Anda merasa tertantang untuk segera mencari jalan keluar *maze*						✓
5.	Anda merasakan suasana misterius yang ditawarkan oleh permainan				✓		
No.	Parameter *Artificial Intelligence*	STS	TS	KS	CS	S	SS
6	Musuh dalam permainan ini dapat menemukan anda					✓	
7	Semakin tinggi level, maka semakin cepat musuh menemukan anda					✓	
8	Tingkat kesulitan musuh sesuai dengan level permainan					✓	
No.	Parameter *Minigame*	STS	TS	KS	CS	S	SS
9	*Minigame* Legot dapat melatih kemampuan logika						✓

b) Halaman Pertama



	anda						
10	*Minigame* Legot pada level *easy* memiliki tingkat kesulitan yang sesuai			√			
11	*Minigame* Legot pada level *medium* memiliki tingkat kesulitan yang sesuai					√	
12	*Minigame* Legot pada level *hard* memiliki tingkat kesulitan yang sesuai					√	
13	*Minigame* Picture Memory dapat melatih memori anda						√
14	*Minigame* Picture Memory pada level *easy* memiliki tingkat kesulitan yang sesuai						√
15	*Minigame* Picture Memory pada level *medium* memiliki tingkat kesulitan yang sesuai						√
16	Minigame Picture Memory pada level *hard* memiliki tingkat kesulitan yang sesuai						√
17	*Minigame* Tap Fast dapat melatih kecepatan refleks anda					√	
18	*Minigame* Tap Fast pada level *easy* memiliki tingkat kesulitan yang sesuai					√	
19	*Minigame* Tap Fast pada level *medium* memiliki tingkat kesulitan yang sesuai					√	
20	*Minigame* Tap Fast pada level *hard* memiliki tingkat kesulitan yang sesuai					√	
No.	Parameter *Gameplay*	STS	TS	KS	CS	S	SS
21	Rancangan *random maze* sudah sesuai dengan tingkat kesulitan levelnya						√
22	Saya merasa *game* ini cocok dimainkan secara *multiplayer*					√	
23	Saya merasa dengan bantuan *power up*, *game* ini menjadi menarik						√
No.	Parameter Kenyamanan	STS	TS	KS	CS	S	SS
24	Aplikasi dapat berjalan lancar tanpa *lag* dan *crash*						√
25	Kontrol pergerakan *player* tidak membingungkan.						√
26	Saya merasa nyaman selama bermain permainan ini						√

C. KRITIK DAN SARAN

.....

.....

.....

BIODATA PENULIS

Vinsensia Sipriana Zega, lahir di Banda Aceh pada tanggal 20 September 1997. Penulis merupakan anak ketiga dari empat bersaudara pasangan Bapak Baza T. Zega dan Ibu Julianar Pandiangan. Penulis menempuh pendidikan formal dimulai dari TK Karya Budi Banda Aceh (2000-2002), SD Karya Budi Banda Aceh (2002-2004), SD RK Serdang Murni Lubuk Pakam (2005-2008), SD Santo Yoseph Pemuda Medan (2005-2008), SMP Santo Yoseph Pemuda Medan (2008-2011), SMA Santo Thomas 1 Medan (2011–2014) dan S1 Informatika ITS (2014-2018). Bidang studi yang diambil oleh penulis pada saat berkuliah di Informatika ITS adalah Interaksi Grafika dan Seni (IGS). Selama masa kuliah, penulis aktif dalam organisasi kemahasiswaan yaitu menjadi staff di bidang Hubungan Luar Himpunan Mahasiswa Teknik Computer-Informatika (2015-2016) dan staff di bidang Media Informasi Badan Eksekutif Mahasiswa Fakultas Teknologi Informasi (2015-2016). Penulis juga aktif dalam berbagai kegiatan kepanitiaan yaitu Schematics 2015, Schematics 2016 dan FTIf Festival 2016. Komunikasi dengan penulis dapat melalui email: **vinsensiasip@gmail.com.**